Федеральное агентство по образованию
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АМГУ»)

УТВЕРЖДАЮ
Зав. кафедрой МАиМ
_____ Т.В. Труфанова
«___»_____ 2007г.

КОМПЬЮТЕРНЫЕ НАУКИ

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

для специальности 010101 – Математика

Составители: Е.М. Салмашова, А.В. Рыженко

Благовещенск
2007

ББК
Т 80

Печатается по решению
редакционно-издательского совета
факультета математики
и информатики
Амурского государственного
университета

Салмашова Е.М., Рыженко А.В.

Учебно-методический комплекс по дисциплине «Компьютерные науки» для студентов очной формы обучения специальности 010101 – «Математика». – Благовещенск. Изд-во Амурский гос. ун-та, 2007. – 261 с.

СОДЕРЖАНИЕ

1. ВЫПИСКА ИЗ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО СТАНДАРТА ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Дисциплина «Компьютерные науки» входит в блок дисциплин федерального компонента для специальности 010101 – «Математика».

Государственный стандарт – ЕН.Ф.01 Компьютерные науки

Понятие информации, общая характеристика процессов сбора, передачи, обработки накопления информации; технические и программные средства реализации информационных процессов. Основные понятия: алгоритм для ЭВМ, базовые конструкции для записи алгоритмов, циклы “для”, “пока”, “если-то-иначе”, выбор, условный и безусловный переход; простейшие типы данных: целый, вещественный, символьный, логический и их представление в ЭВМ; массивы данных; организация ввода и вывода; понятие о файловой системе; файлы последовательного доступа и прямого доступа; форматный и бесформатный ввод/вывод; простейшие алгоритмы обработки данных: вычисление по формулам, последовательный и бинарный поиск, сортировка, итерационные алгоритмы поиска корней уравнений, индуктивная обработка последовательностей данных, рекуррентные вычисления.

Структуры данных: вектор, матрица, запись (структура), стек, дек, очередь, последовательность, список, множество, бинарное дерево; реализация структур данных на базе линейной памяти ЭВМ; непрерывный и ссылочный способы реализации структур данных; реализации множества (битовая, непрерывная, хеш-реализация); алгоритмы обработки коллизий в хеш-реализации.

Рекурсивные и итерационные алгоритмы обработки данных; условия, обеспечивающие завершение последовательности рекурсивных вызовов; идеи реализации рекурсивных вызовов в подпрограммах; инвариантная функция и инвариант цикла; взаимосвязь итерации и рекурсии, индуктивное вычисление функций на последовательности данных.

Структуры данных в прикладных программах; примеры использования и реализации различных структур (редактор текстов, стековой калькулятор); принципы построения файловых систем; каталог, таблица размещения файлов, распределение блоков файла по диску.

Компиляция и интерпретация: основные этапы компиляции, лексический, семантический анализ выражения, формальная грамматика, компилятор формулы, дерево синтаксического разбора.

Понятие об операционной системе: процесс, состояние процесса, прерывание, планирование процессов, понятие о тупиках и способах их устранения.

Надежность программного обеспечения: методы тестирования и отладки программ, переносимость программ, технология программирования, принципы создания пакетов стандартных программ, принципы обеспечения дружественного интерфейса прикладных программ.

Понятие об архитектуре ЭВМ: процессор и система его команд, структура памяти ЭВМ и способы адресации, выполнение команды в процессоре, взаимодействие процессора, памяти и периферийных устройств.

Локальные и глобальные сети ЭВМ; основы защиты информации и сведений, составляющих государственную тайну; методы защиты информации.

Компьютерный и вычислительный практикум: реализация алгоритмов обработки данных, возникающих в задачах алгебры, математического анализа, математической статистики, задач обработки изображений, задачах линейного программирования; сети и работа в них.

2. РАБОЧАЯ ПРОГРАММА

по дисциплине «Компьютерные науки»
для специальности 010101 – «Математика»

Курс 1,2,3 Семестр 1-6
Лекции 108 (час.) Зачет 3,4,5,6 семестр
Экзамен 1,2 семестр
Практические занятия нет
Лабораторные работы 252 (час.)
Самостоятельная работа 240 (час.)
Всего часов 600 (час.)

1. Цели и задачи дисциплины, ее место в учебном процессе

1.1. Цели и задачи курса

Основной целью дисциплины является обучение студентов основным навыкам работы на ЭВМ, знакомство с системным и прикладным программным обеспечением, обучение основам программирования, теоретическим основам и практическим навыкам проектирования и реализации программ на современных ЭВМ.

Курс «Компьютерные науки» является основой для изучения широкого круга дисциплин инженерного цикла, использующих вычислительную технику. Курс соответствует требованиям подготовки студентов соответствующей специальности.

1.2. Требования к уровню освоения содержания дисциплины

В результате изучения дисциплины студенты должны знать и уметь использовать:

1) программирование и использование возможностей вычислительной техники и программного обеспечения;
2) системные программные средства, операционные системы, оболочки, сервисные программы;
3) современные методы и средства программирования;
4) современные языки программирования;
5) возможности современных информационных технологий.

Студенты должны иметь опыт:

1) программирования на основных алгоритмических языках;
2) использования стандартного программного обеспечения, пакетов программ общего назначения и утилит;
3) работы с современными информационными технологиями и сетями.

По окончанию курса студент должен иметь представление об:

1) основных этапах решения задач на ЭВМ;
2) информации, общих характеристиках процессов сбора, передачи, обработки и накопления информации;

3) технических и программных средствах реализации информационных процессов;
4) алгоритмизации и программировании;
5) языках программирования; методах и основных этапах трансляции.
6) современных информационных технологиях и сетях;
7) современных способах конструирования программ;
8) синтаксисе, семантике, формальных способах описания языков;

1.3. Перечень дисциплин с указанием разделов (тем), усвоение которых студентами необходимо при изучении данной дисциплины

Так как дисциплина изучается в первом семестре первого курса, то она базируется на знаниях, полученных в рамках школьной программы. А далее на знаниях, полученных в рамках данной дисциплины на ранних курсах.

2. Содержание дисциплины

2.1. Наименование тем, их содержание, объем в лекционных часах

ТЕМАТИЧЕСКИЙ ПЛАН ЛЕКЦИОННЫХ ЗАНЯТИЙ

	Наименование темы	Кол-во часов
1	**1 семестр** Вычислительная техника в работе инженера	2
2	Информация, способы представления информации.	2
3	Технические и программные средства реализации информационных процессов**.**	4
3	Системное и прикладное программное обеспечение. Языки программирования	2
4	Системное программное обеспечение	4
5	Прикладное программное обеспечение	4
6	Информационные базы данных. Системы управления базами данных	4
7	Современные информационные технологии и сети	2
8	Алгоритм и алгоритмизация	2
9	Этапы и методы разработки программ	2
10	Основы программирования на языке Паскаль	8
ИТОГО за 1 семестр		36 часов
11	**2 семестр** Программирование на языке Паскаль	20
12	Работа с графикой в среде Турбо Паскаль	16
ИТОГО за 2 семестр		36 часов
13	**3 семестр** Модульное программирование	4
14	Работа с динамическими структурами данных	6
15	Объектно-ориентированное программирование	8

ИТОГО за 3 семестр		18 часов
16	**4 семестр** Система визуального объектно-ориентированного программирования Delphi	4
17	Определение собственных типов данных	2
18	Компоненты страниц Standart	4
19	Компоненты страницы Additional	2
20	Графические возможности	2
21	Страница Dialogs	2
22	Отладка программ	2
ИТОГО за 4 семестр		18 часов
ИТОГО		108 часов

1 семестр (36 часов)

Тема 1. Вычислительная техника в работе инженера.

Предмет, цели и задачи курса. Вычислительная техника и научно-технический прогресс. Представление об информационном обществе. Информационные ресурсы, продукты и услуги. ЭВМ, их назначение и использование в различных областях человеческой деятельности.

Представление об информационном обществе. Информационные ресурсы, продукты и услуги.

Тема 2. Информация, способы представления информации.

Понятие информации и кодирования. Виды и свойства информации. Этапы обработки информации в ЭВМ. Общая характеристика сбора, хранения, обработки, накопления и передачи информации. Способы представления и преобразования информации. Системы счисления.

Тема 3. Технические и программные средства реализации информационных процессов.

ЭВМ как универсальное устройство обработки информации, классификации ЭВМ. Структура и функции аппаратной части ПК. Микропроцессоры, запоминающие устройства, основные внешние устройства ПК. Тенденции развития ЭВМ.

Тема 4. Системное программное обеспечение.

Программное обеспечение ЭВМ, его состав и назначение. Языки и системы программирования. Понятие операционной системы, назначение. Операционная система MS-DOS. Программы-драйверы, программы - оболочки. Операционные оболочки, оболочка NC,TC. Утилиты. Основные понятия и основы работы с Windows-XP.

Тема 5. Прикладное программное обеспечение.

Классификация прикладных программ. Текстовые редакторы, назначение и классификация, общие сведения о создании и редактировании текстов. WORD, функциональные возможности.

Табличные процессоры (EXCEL), принципы их построения и обработки. Графические редакторы. Понятие графической информации,

способы ее представления и обработки в ЭВМ. Пакеты программ, ориентированные на работу с графической информацией, назначение и основные возможности. Системы деловой и научной графики.

Тема 6. Информационные базы данных. Системы управления базами данных.

Типы данных. Способы представления данных в ЭВМ, модели данных. Понятие базы данных, их назначение, классификации и использование. Принципы работы с базами данных (Access).

Тема 7. Современные информационные технологии и сети.

Понятие компьютерных сетей, общие принципы организации и функционирования. Архитектура сетей. Локальные и глобальные компьютерные сети. Интернет. Понятие информационных технологий и систем, классификации, назначение, области применения. ГИС-технологии, назначение и основные возможности. Экспертные системы.

Тема 8. Алгоритм и алгоритмизация.

Алгоритм, способы задания и описания алгоритмов. Основные конструкции алгоритмов. Типы алгоритмов. Организация алгоритмов линейной, разветвляющейся, циклической и сложной комбинированной структуры.

Тема 9. Этапы и методы разработки программ.

Этапы внутреннего и внешнего проектирования. Отладка и тестирование программ. Модели надежности.

Тема 10. Основы программирования на языке Паскаль.

Общая характеристика языка. Основные конструкции языка. Понятие блочной структуры. Синтаксические диаграммы. Специальные символы, зарезервированные слова, идентификаторы, числа, строки, комментарии. Типы данных. Описание типов. Структура программы. Константы и переменные. Описание констант, переменных, меток. Операции и выражения. Приоритет операций и выражений. Арифметические операции, операции отношения, булевские (логические) операции, битовые булевские и сдвиговые операции, операция взятия адреса. Операторы. Простые операторы (присваивания, перехода, оператор процедуры). Структурные операторы. Ввод-вывод данных. Линейные, разветвляющие и циклические конструкции, их реализация на Паскале. Процедуры и функции. Структура процедур и функций. Область действия (сфера видимости) идентификаторов переменных при использовании процедур и функций. Локальные и глобальные переменные. Формальные и фактические параметры в процедурах и функциях. Вызов процедур и функций. Процедурные типы. Массивы. Одномерные, двухмерные массивы. Организация записи данных в массив и чтения данных из массива. Вывод массива на печать. Сортировка массивов. Методы прямого выбора, вставки и пузырька. Сравнение прямых методов сортировки.

2 семестр (36 часов)

Тема 11. Программирование на языке Паскаль.

Понятие рекурсии и основные определения. Формы рекурсивных процедур (выполнение действий на рекурсивном спуске, на рекурсивном возврате). Структурированные типы данных: строки, множества, записи, записи с вариантами. Операции со строками (сцепление строк, добавление в строку и удаление из строки подстроки, поиск подстроки в строке, преобразование строки в числовое представление). Функции для работы со строками. Описание множественного типа в Паскале. Математические операции для работы с множествами (объединение, пересечение, разность, включение подмножества во множество, принадлежность элемента множеству) и их программная реализация. Записи. Фиксированные и вариантные записи. Операции над записями (заполнение полей записи, чтение нужного поля записи, сравнение полей). Оператор присоединения WITH. Файлы в Паскале. Типизированные файлы. Процедуры и функции для работы с типизированными файлами. Текстовые файлы. Процедуры и функции для работы с текстовыми файлами. Файлы без типа. Процедуры и функции для работы с нетипизированными файлами.

Тема 12. Работа с графикой в среде Турбо Паскаль.

Работа в текстовом видеорежиме. Константы цвета. Окна в текстовом видеорежиме. Прямой доступ к видеопамяти. Работа в графическом видеорежиме. Аппаратная и программная поддержка графики, инициализация графики. Базовые процедуры и функции: построения графических образов, работы с цветовой палитрой, работы с текстом. Константы цвета, типа линий и их толщины. Константы типа шрифта и выравнивания текста (по горизонтали и по вертикали). Предопределенные типы: тип установки цветов палитры, тип установки вида линий, тип установки оформления текста, тип установки вида закраски, тип для задания координат точек на экране. Графические драйверы и режимы. Инициализация графического режима. Ошибки инициализации графического режима и их обработка. Построение графиков функций. Построение графических примитивов. Создание динамических образов. Работа со звуком.

3 семестр (18 часов)

Тема 13. Модульное программирование.

Стандартные модули и создаваемые пользователем. Структура TP-модуля. Заголовок модуля, интерфейс модуля, исполнительная часть модуля, секция инициализации. Использование модуля в основной программе. Стандартные модули Crt, Graph, System, Dos, WinDos, String, Overlay, Printer. Динамически связываемые библиотеки (DDL), импорт процедур и функций из DDL.

Тема 14. Работа с динамическими структурами данных.

Указатели и ссылочные типы данных. Выделение и освобождение динамической памяти. Процедуры и функции, используемые при работе с указателями и динамической памятью. Динамические массивы. Сортировка динамических массивов. Улучшенные методы сортировки массивов. Быстрая сортировка (сортировка Хора), сортировка Шелла. Очередь. Создание очереди, добавление и удаление элемента очереди. Стек. Создание стека.

Добавление и удаление элемента стека. Кольцевой список. Мультисписки. Деревья. Бинарное дерево. Бинарная сортировка массивов. Хэш - таблицы и хэш-функции. Сортировка массивов при помощи хэш-таблиц. Использование хэш-таблиц и хэш-ключей для кодирования информации.

Тема 15. Объектно-ориентированное программирование.

Основные принципы объектно-ориентированного программирования. Понятие объекта, описание объекта в TP7. Инкапсуляция. Методы. Классы и объекты. Сообщения и события, реагирование объекта на сообщения и события. Наследование. Полиморфизм. Иерархия объектов. Виртуальные методы. Объявление виртуальных методов. Возможности модификации программы при использовании виртуальных методов. Конструкторы и деструкторы. Понятие об основных функциях Windows API.

4 семестр (18 часов)

Тема 16. Система визуального объектно-ориентированного программирования Delphi.

Назначение и основные функции пакета Delphi. Основы ООП в Delphi. Язык программирования Object Pascal. Основные понятия Delphi: форма, окно редактирования кода, инспектор объектов, проект.

Тема17 . Определение собственных типов данных.

Описание нового типа. Перечисляемые типы. Типы поддиапазонов. Структурные типы данных. Указатели. Варианты. Сложные структуры данных. Основные стандартные функции для работы с типами. Преобразование типов. Инициализация констант сложных типов.

Тема 18. Компоненты страниц Standart.

Tframe- рама и шаблоны компонентов. TmainMenu- главное меню формы. TpopupMenu- контекстное меню. Tlabel- метка для отображения текста. TEdit-ввод и отображение текста. Tmemo- ввод и отображение многострочного текста. TButton- командная кнопка. TcheckBox- независимый переключатель (флажок). TradioButton - зависимый переключатель (радиокнопка). TlistBox-список выбора. TcomboBox- раскрывающийся список выбора. TscrollBar- полоса прокрутки. TgroupBox - панель группирования. TradioGroup- панель радиокнопок. Tpanel - панель. События и реакция на них. Обработка щелчка мыши.

Тема 19. Компоненты страницы Additional.

TbitBtn- командная кнопка с изображением. TspeedButton- кнопка для инструментальных панелей. TmaskEdit- маска ввода. TstringGrid- таблица строк. TdrawGrid- произвольная таблица. Tbevel- кромка. TscrollBox- панель с прокруткой. TcheckListBox- группа независимых переключателей. Tsplitter -компонент для изменения размеров. TstatieText- метка для отображения текста. TcontrolBar- инструментальная панель.

Тема 20. Графические возможности.

Класс Tfont. Класс Tpen и Tbrush. Класс Tcanvas. Компонент Timage - отображение картинок. Tshare - стандартная фигура. Tchart- построитель графиков.

Тема 21. Страница Dialogs.

TopenDialog и TsaveDialog - диалоги открытия и сохранения файлов. TopenPictureDialog и TsavePictureDialog- диалоги открытия и сохранения графических файлов. TfontDialog - диалог выбора шрифта. TcolorDialog -диалог выбора цвета. TfindDialog - диалог поиска и замены.

Тема 22.Отладка программ.

Что такое отладка. Причины ошибок. Синтаксические ошибки. Логические ошибки. Выполнение по шагам. Просмотр значений. Просмотр и изменение значений. Просмотр и анализ кода. Расширенные средства отладки. Прерывание по условию. Организация точек прерывания в группы. Ведение протокола работы. Инспектор отладки. Исключительные ситуации. Стандартные классы исключительных ситуаций. Программный обработчик ошибок.

2.2. Лабораторные занятия, их содержание и объем в часах

ТЕМАТИЧЕСКИЙ ПЛАН ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Наименование темы		Кол-во часов
1	**1 семестр** Знакомство с ПК. Изучение операционной системы MS DOS. Программная оболочка NC (TC). Работа с файлами и каталогами. Утилиты NC (NU). Архивы	4
2	Операционная система Windows	2
3	Текстовой редактор MS WORD	4
4	Табличный процессор EXCEL	6
5	Базы данных ACCESS	6
6	Компьютерная графика. Пакеты Corel, Photoshop	6
7	Изучение среды программирования Турбо Паскаль. Структура программы на Паскале. Линейные алгоритмы	2
8	Программирование формул. Программы разветвляющей структуры. Табулирование функций	2
9	Программы циклической структуры. Нахождение суммы ряда с точностью. Решение уравнений.	4
10	Вложенные циклы	2
11	Вычисление интеграла	2
12	Работа с массивами	2
13	Сортировка массивов	4
14	Обработка матриц	4
ИТОГО за 1 семестр		54 часа
15	**2 семестр** Рекурсия на Паскале	2
16	Решение задач с использованием подпрограмм	4
17	Строки	2
18	Множества	2
19	Записи. Записи с вариантами	6

20	Файлы в Паскале	6
21	Текстовые файлы	2
22	Работа с текстом	2
23	Работа с графикой в среде Турбо Паскаль. Инициализация графики	2
24	Построение графиков функций	4
25	Построение графических примитивов	4
ИТОГО за 2 семестр		36 часов
26	**3 семестр** Создание пользовательского модуля	4
27	Указатели и ссылочные типы	4
28	Работа с динамическими массивами (создание массива, поиск элемента в массиве, сортировка массива)	4
29	Сортировка Хора (быстрая сортировка)	2
30	Сортировка Шелла	2
31	Очередь	4
32	Стек	4
33	Бинарное дерево	2
34	Хэш-таблицы	4
35	Кодирование информации с помощью хэш-ключей	2
36	Объекты. Создание графического объекта (кубика)	2
37	Наследование	2
ИТОГО за 3 семестр		36 часов
38	**4 семестр** Среда Delphi	4
39	Компоненты страницы Standart	4
40	Компоненты страницы Additional	4
41	Страница Dialogs	4
42	Работа с массивами в Delphi (описание, ввод массива, сортировка массива)	4
43	Работа с графикой в Delphi	4
44	Построение графиков в Delphi	4
45	Диалоговые окна в Delphi	4
46	Отладка программ в Delphi	4
ИТОГО за 4 семестр		36 часов
47	**5 семестр** MATLAB и Mathematica	
48	MATLAB – возможности системы, назначение, направление использования. Переменные.	2
49	Построение графиков. Графические возможности.	2
49	Работа с матрицами и векторами. Полиномы.	6
50	Решение уравнений. Итерации. Решение СЛАУ.	8
51	Основы программирования в среде MATLAB. Встроенные функции пакета.	6

52	Тулбоксы и их функции.	2
53	Контрольное занятие.	2
54	Основные возможности пакета Mathematica 3/5.	6
55	Символьные вычисления.	4
56	Встроенная графика.	4
57	Процедурное программирование.	4
58	Ввод и вывод данных.	4
59	Зачетное занятие.	4
ИТОГО за 5 семестр		54 часа
60	**6 семестр** MATHCAD	
61	MATHCAD. Основные возможности пакета. Создание и редактирование формул. Создание математических и текстовых областей, параграфы.	2
62	Символьные вычисления. Графические возможности. Создание различного рода графиков, их форматирование и использование.	2
63	Форматирование выражений и результатов. Локальный и глобальный формат. Дискретный аргумент. Переменные и константы. Способы определения и вычисления переменных и функций. Встроенные переменные.	2
64	Создание векторов и матриц, способы их отображения. Оператор векторизации. Основные встроенные функции. Решение СЛАУ.	4
65	Аппроксимация функций.	4
66	Решение алгебраических уравнений.	2
67	Программирование.	4
68	Решение дифференциальных уравнений и систем.	2
69	Ряды Фурье.	2
70	Задачи из теории вероятностей.	2
71	Решение задач численными методами.	6
72	Зачетное занятие.	4
ИТОГО за 6 семестр		36
ИТОГО		252 часа

При выполнении лабораторных работ по данному курсу студенты должны ознакомиться с современными методами и средствами программирования, языками программирования; стандартным программным обеспечением, пакетами программ общего назначения и утилит; с современными информационными технологиями и сетями; языком Турбо Паскаль, Delphi, типах данных, способах и механизмах управления данными, методах и основных этапах трансляции; продемонстрировать умение программирования различных функциональных и вычислительных задач.

Лабораторная работа выполнятся строго в соответствии с выданным преподавателем заданием и вариантом. По окончанию занятия студент

обязан сдать разработанную программу и объяснить алгоритм решения поставленной задачи.

2.3. Самостоятельная работа студентов

Для студентов специальности 010101 – «Математика» на самостоятельную работу по рабочей программе отводиться 240 часов.

В качестве самостоятельной работы по дисциплине «Компьютерные науки» студентам предлагается рассмотреть и изучить следующие вопросы:

1. Работа с текстовыми, табличными редакторами, базами данных – основные возможности.
2. Глобальные и локальные сети. Работа в Интернет, основные услуги, предоставляемые сетями. Посещение Интернет-центра.
3. Среда программирования Турбо Паскаль. Функциональные возможности меню.
4. Численные методы решения уравнений (половинного деления, касательных, Ньютона, простой итерации).
5. Приближенное интегрирование функции (метод прямоугольников, метод трапеций).
6. Типы данных (стек, очередь).
7. Стандартные модули Crt, Graph, System, Dos, WinDos, String, Overlay, Printer.
8. Сортировка массивов при помощи хэш-таблиц.
9. Понятие об основных функциях Windows API.
10. Delphi: основные стандартные функции для работы с типами. Преобразование типов.
11. События и реакция на них.
12. Компонент Timage - отображение картинок.

Контроль за выполнением самостоятельной работы осуществляется проведением аудиторных самостоятельных работ, а также вопросы предложенные для самостоятельного изучения включены в перечень экзаменационных вопросов.

2.4. Вопросы к зачетам и экзаменам

1 семестр - экзамен

1. Предмет, цели и задачи курса «Компьютерные науки». Вычислительная техника и научно-технический прогресс.
2. ЭВМ, их назначение и использование в различных областях человеческой деятельности.
3. Использование ЭВМ в учебном процессе.
4. Представление об информационном обществе.
5. Информационные ресурсы, продукты и услуги.
6. Понятие информации, виды и свойства информации.
7. Этапы обработки информации в ЭВМ. Общая характеристика сбора, хранения, обработки, накопления и передачи информации.
8. Способы представления и преобразования информации.

9.Системы счисления.
10.ЭВМ как универсальное устройство обработки информации, классификации ЭВМ.
11.Структура и функции аппаратной части ПК. Микропроцессоры, запоминающие устройства, основные внешние устройства ПК.
12.Тенденции развития ЭВМ.
13.Программное обеспечение ЭВМ, его состав и назначение.
14.Понятие операционной системы, назначение. Операционная система MS DOS.
15.Программы-драйверы, программы - оболочки. Операционные оболочки, оболочка NC, TC.
16.Утилиты, NU.
17.Основные понятия и основы работы с Windows XP.
18.Классификация прикладных программ.
19.Текстовые редакторы, назначение и классификация, общие сведения о создании и редактировании текстов.
20.WORD, функциональные возможности.
21.Табличные процессоры (EXCEL), принципы их построения и обработки.
22.Графические редакторы. Понятие графической информации, способы ее представления и обработки в ЭВМ.
23.Пакеты программ, ориентированные на работу с графической информацией, назначение и основные возможности.
24.Системы деловой и научной графики.
25.Понятие базы данных, их назначение, классификации и использование.
26.Принципы работы с базами данных (Access).
27.Понятие компьютерных сетей, общие принципы организации и функционирования.
28.Архитектура сетей. Локальные и глобальные компьютерные сети.
29.Интернет.
30.Понятие информационных технологий и систем, классификации, назначение, области применения.

2 семестр – экзамен

1.Современные ЭВМ, их классификации, назначение и использование в инженерной практике.
2.Программное обеспечение ЭВМ. Системное и прикладное программное обеспечение.
3.Операционные системы. Языки программирования.
4.Технические средства ЭВМ.
5.Представление информации в ЭВМ. Системы счисления.
6.Алгоритмы, способы задания и описания. Основные конструкции алгоритмов.
7.Константы: целые, вещественные, логические, символьные. Понятие идентификатор.

8.Переменные, их описание в программе. Скалярные типы. Тип integer. Операции и функции, применимые для данного типа.
9.Переменные, их описание в программе. Скалярные типы. Тип real. Операции и функции, применимые для данного типа.
10.Переменные, их описание в программе. Скалярные типы. Тип boolean. Логические операции AND, OR, NOT, XOR.
11.Переменные, их описание в программе. Скалярные типы. Тип char. Операции и функции, применимые для данного типа.
12.Переменные, их описание в программе. Скалярные типы. Перечисляемые типы. Операции и функции, применимые для данного типа.
13.Переменные, их описание в программе. Скалярные типы. Ограниченный тип. Операции и функции, применимые для данного типа.
14.Структура программы на языке Паскаль. Комментарии.
15.Оператор присваивания. Приоритет операций.
16.Операторы WRITE, WRITELN.
17.Оператор безусловного перехода COTO.
18.Составной оператор.
19.Оператор условного перехода IF.
20.Оператор CASE. Операции сравнения.
21.Оператор цикла WHILE.
22.Оператор цикла REPEAT.
23.Оператор цикла FOR.
24.Процедура ввода READ, READLN.
25.Функции SUCC(x), PRED(x), ORD(x), ODD(x), ROUND(x), TRUNC(x).
26.Массивы. Множества.
27.Подпрограммы общего вида (PROCEDURE).
28.Подпрограммы-функции (FUNCTION).
29.Рекурсия. Рекурсивные программы.
30.Локальные и глобальные переменные.
31.Формальные и фактические параметры в процедуре.
32.Понятие о файлах. Процедуры открытия и закрытия файлов.
33.Процедуры и функции работы с файлами.
34.Текстовые файлы.
35.Записи. Записи с вариантами.
36.Процедурные типы.
37.Нетипизированные параметры процедур и функций.
38.Работа с графикой в среде Turbo Pascal.
39.Видеорежимы. Процедуры и функции, предназначенные для управления графическими режимами.
40.Загрузка драйверов и шрифтов в программу.
41.Процедуры и функции, предназначенные для построения графических образов.

42.Процедуры и функции, предназначенные для работы с цветовой палитрой.
43.Процедуры и функции, предназначенные для вывода текстовой информации.
44.Процедуры и функции, предназначенные для работы с графическими образами.

3 семестр – зачет

1.Основы модульного программирования. Структура TP-модуля.
2.Стандартные модули. Модуль Crt.
3.Модуль Graph.
4.Модули System, Dos, Windows.
5.Создание пользовательского модуля и использование его в программе.
6.Структурированные типы данных. Множества.
7.Динамические данные. Указатели.
8.Ссылочные типы.
9.Динамические массивы.
10.Списки.
11.Стек. Организация стека.
12.Операции со стеками. Удаление и добавление элемента в стек.
13.Выделение и освобождение динамической памяти. Процедуры и функции, используемые при работе с указателями и динамической памятью.
14.Очередь. Организация очереди.
15.Удаление первого и добавление последнего элемента в очередь.
16.Принципы объектно-ориентированного программирования.
17.Описание объекта.
18.Инкапсуляция. Методы. Классы и объекты.
19.Сообщения и события, реагирование объекта на сообщения и события.
20.Наследование.
21.Полиморфизм.
22.Иерархия объектов.
23.Виртуальные методы. Объявление виртуальных методов.
24.Конструкторы и деструкторы.
25.Понятие о функциях Windows API.

4 семестр – зачет

1.Структурные типы данных в Delphi.
2.Описание нового типа.
3.Сложные структуры данных.
4.Основные стандартные функции для работы с типами.
5.Преобразование типов.
6.Инициализация констант сложных типов.
7.Структура подпрограмм.
8.Понятие объекта.
9.Понятие класса.
10.Типы методов.

11.Динамическое конструирование объектов.
12.Использование визуальных компонентов.
13.Компонент Меню.
14.Компонент Контекстное меню.
15.Компонент Текстовая область.
16.Компонент Переключатель.
17.Компонент Группа переключателей.
18.Компонент Поле со списком.
19.Компонент Флажок.
20.Компонент Полоса прокрутки.
21.Иерархия компонентов.
22.Класс Tobject.
23.Класс Tpresistent (наследник Tobject).
24.Класс Tcomponent (наследник Tpresistent).
25.Форма.
26.Управление проектом. Добавление новой формы.
27.Что такое отладка программы?
28.Причины ошибок.
29.Синтаксические ошибки.
30.Логические ошибки.
31.Выполнение программы по шагам.
32.Просмотр и изменение значений.
33.Просмотр и анализ кода.
34.Прерывания по условию.
35.Организация точек прерывания в группы.
36.Ведение протокола работы.
37.Отладка внешних процессов.
38.Инспектор объектов.
39.Генерация исключительных ситуаций.
40.Стандартные классы исключительных ситуаций.
41.Контроль над исключительными ситуациями.
42.Программный обработчик ошибок.

5 семестр – зачет

1.Переменные и их типы. Команды для задания различного вида матриц.
2.Построение графиков функций, виды графиков.
3.Нахождение корней уравнения, полинома.
4.Многомерные массивы: определение, команды обработки.
5.Программирование: m-файлы и m-функции.
6.Решение уравнений, задач мат. анализа и линейной алгебры с помощью тулбоксов.
7.Визуализация в пакете Mathematica.
8.Символьные вычисления. Преобразование полиномов.
9.Символьные решение алгебраических и трансцендентных уравнений.
10.Математический анализ. Решение дифференциальных уравнений.
11.Встроенная графика.

12.Программирование: составные операторы, условные операторы, условные циклы.
13.Ввод и вывод данных в файлы и в другие программы.

6 семестр – зачет

1.MATHCAD. Основные возможности пакета. Создание и редактирование формул. Создание математических и текстовых областей, параграфы.
2.Символьные вычисления. Графические возможности. Создание различного рода графиков, их форматирование и использование.
3.Форматирование выражений и результатов. Локальный и глобальный формат. Дискретный аргумент. Переменные и константы. Способы определения и вычисления переменных и функций. Встроенные переменные.
4.Создание векторов и матриц, способы их отображения. Оператор векторизации. Основные встроенные функции. Решение СЛАУ.
5.Аппроксимация функций.
6.Решение алгебраических уравнений.
7.Программирование.
8.Решение дифференциальных уравнений и систем.
9.Ряды Фурье.
10.Задачи из теории вероятностей.
11.Решение задач численными методами.
12.Программирование: составные операторы, условные операторы, условные циклы.
13.Ввод и вывод данных в файлы и в другие программы.

2.5. Перечень и темы промежуточных форм контроля знаний студентов

Текущий контроль за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения лабораторных работ посредством устного опроса по изучаемым разделам и проверки отчетности по лабораторным работам и индивидуальным занятиям в часы консультаций. Промежуточный контроль предполагает систематическое проведение контрольных работ.

2.6. Требования к знаниям студентов, предъявляемые на экзамене и зачете

Экзамен или зачет сдаются в конце семестра. Форма сдачи – устная. Необходимым допуском на экзамен (зачет) является сдача всех лабораторных работ, положительные оценки за промежуточные контрольные работы. В предлагаемый билет входят три вопроса: два теоретических и один практический (разработка программы), на которые студент должен дать развернутый ответ. Показать полное знание теории по данной части курса, продемонстрировать свободную ориентацию в материале, знание понятий и терминологии, ответить на дополнительные вопросы. Выполнение указанных требований оценивается оценкой (зачтено – не зачтено).

3. Учебно-методические материалы по дисциплине

3.1. Перечень обязательной (основной) литературы

1. Абрамов С.А., Зима В.С. Начало программирования на языке Паскаль.-М: Наука, 1999.- 112с.
2. Бобровский С. Delphi 5: учебный курс. - СПб: Издательство «Питер», 2000.-640 с.
3. Вирт Н. Алгоритмы + структура данных = программы: перевод с англ.-М.Мир,1998.- 406с.
4. Емелина Е.И. Основы программирования на языке Паскаль.-М.: Финансы и статистика, 1997.- 208с.
5. Зуев Е.А. Программирование на языке Turbo Pascal 6.0, 7.0.-М: Радио и связь, Веста, 1997.-380с.
6. Королев Л. Н., Миков А. И. Информатика. Введение в компьютерные науки.-М.:Высшая школа, 2003.
7. Сафронов И. К. Задачник-практикум по информатике.-СПб.:BHV-СПб, 2002.
8. Семакин И.Г., Хеннер Е.К. Информатика. Задачник-практикум.-М.: Лаборатория Базовых Знаний, Том 2, 2002.

3.2. Перечень дополнительной литературы

1. Гусева А.И. Учимся информатике. Задачи и методы их решения. – М: Диалог-Мифи, 2001.
2. Левин А. Самоучитель работы на компьютере. – М.:Международное агенство A.D.&T.,1997 – 480с.
3. Могилев А. В., Пак Н. И., Хеннер Е. К. Информатика.-М.: Академия, Серия: Высшее образование ("Академия"), 2004. Е. К.
4. Фигурнов В. IBM PC для пользователя.-С.Петербург,1997г.
5. Франклен Г. MS DOS 5.0 для пользователя.- Киев: Торгово-издательское бюро BHV, 1992.

3.3. Перечень методических пособий

1. Гордиенко А.М., Рогов Д.К. Численные методы проектирования на ЭВМ./ Методические указания, М., 1990. – 31с.
2. Макарчук Т.А. Основы программирования в Delphi 5. Учебно-методическое пособие. -Благовещенск: Амурский гос.ун-т.2001-66с.
3. Ракитин В.И., Первушин В.Е. Практическое руководство по методам вычислений. – М.: Выс.шк., 1998. – 383с.
4. Решетнева Т.Г. Алгоритмизация и программирование в среде MathCAD: Практикум.-Благовещенск: Амурский гос.ун-т.2002-79с.
5. Чалкина Н.А., Масленко Т.В. Практикум по решению задач на языке Turbo Pascal 7.0 по курсу «Информатика»/.-Благовещенск: Амурский гос.ун-т.1999-110с.

3. КРАТКИЙ КОНСПЕКТ ЛЕКЦИЙ

***ТЕМА:* ЭВМ как средство обработки информации**

При рассмотрении ЭВМ как средства обработки информации важную роль играют понятие архитектуры ЭВМ, классификация ЭВМ, структура и принципы функционирования ЭВМ, а также основные характеристики вычислительной техник.

Понятие архитектуры ЭВМ

С середины 60-х годов существенно изменился подход к созданию вычислительных машин. Вместо независимой разработки аппаратуры и некоторых средств математического обеспечения стала проектироваться система, состоящая из совокупности *аппаратных (hardware)* и *программных (software)* средств. При этом на первый план выдвинулась концепция их взаимодействия. Так возникло принципиально новое понятие — архитектура ЭВМ.

Под ***архитектурой ЭВМ*** понимается совокупность общих принципов организации аппаратно-программных средств и их характеристик, определяющая функциональные возможности ЭВМ при решении соответствующих классов задач.

Архитектура ЭВМ охватывает широкий круг проблем, связанных с построением комплекса аппаратных и программных средств и учитывающих множество факторов. Среди этих факторов важнейшими являются: стоимость, сфера применения, функциональные возможности, удобство эксплуатации, а одним из главных компонентов архитектуры являются аппаратные средства. Основные компоненты архитектуры ЭВМ можно представить в виде схемы, показанной на рис. 2.

Рис. 2. Основные компоненты архитектуры ЭВМ

Архитектуру вычислительного средства следует отличать от его структуры. Структура вычислительного средства определяет его конкретный состав на некотором уровне детализации (устройства, блоки узлы и т. д.) и описывает связи внутри средства во всей их полноте. Архитектура же определяет правила взаимодействия составных частей вычислительного средства, описание которых выполняется в той мере, в какой это необходимо для формирования правил их взаимодействия. Она регламентирует не все

связи, а наиболее важные, которые должны быть известны для более грамотного использования данного средства.

Так, пользователю ЭВМ безразлично, на каких элементах выполнены электронные схемы, схемно или программно реализуются команды и т. д. Важно другое: как те или иные структурные особенности ЭВМ связаны с возможностями, предоставляемыми пользователю, какие альтернативы реализованы при создании машины и по каким критериям принимались решения, как связаны между собой характеристики отдельных устройств, входящих в состав ЭВМ, и какое влияние они оказывают на общие характеристики машины. Иными словами, архитектура ЭВМ действительно отражает круг проблем, относящихся к общему проектированию и построению вычислительных машин и их программного обеспечения.

Классификация ЭВМ

Чтобы судить о возможностях ЭВМ, их принято разделять на группы по определенным признакам, т. е. классифицировать. Сравнительно недавно классифицировать ЭВМ по различным признакам не составляло большого труда. Важно было только определить признак классификации, например: по назначению, по габаритам, по производительности, по стоимости, по элементной базе и т. д.

С развитием технологии производства ЭВМ классифицировать их стало все более затруднительно, ибо стирались грани между такими важными характеристиками, как производительность, емкость внутренней и внешней памяти, габариты, вес, энергопотребление и др. Например, персональный компьютер, для размещения которого достаточно стола, имеет практически такие же возможности и технические характеристики, что и достаточно совершенная в недавнем прошлом ЭВМ Единой системы (ЕС), занимающая машинный зал в сотни квадратных метров. Поэтому разделение ЭВМ по названным признакам нельзя воспринимать как классификацию по техническим параметрам. Это, скорее, эвристический подход, где большой вес имеет предполагаемая сфера применения компьютеров.

С этой точки зрения ***классификацию*** вычислительных машин по таким показателям, как ***габариты и производительность,*** можно представить следующим образом:

- сверхпроизводительные ЭВМ и системы (супер-ЭВМ);
- большие ЭВМ (универсальные ЭВМ общего назначения);
- средние ЭВМ;
- малые или мини-ЭВМ;
- микро-ЭВМ;
- персональные компьютеры;
- микропроцессоры.

Отметим, что понятия «большие», «средние» и «малые» для отечественных ЭВМ весьма условны и не соответствуют подобным категориям зарубежных ЭВМ.

Исторически первыми появились **большие ЭВМ (универсальные ЭВМ общего назначения)**, элементная база которых прошла путь от электронных

ламп до схем со сверхвысокой степенью интеграции. В процессе эволюционного развития больших ЭВМ можно выделить отдельные периоды, связываемые с пятью поколениями ЭВМ.

Поколение ЭВМ определяется элементной базой (лампы, полупроводники, микросхемы различной степени интеграции), архитектурой и вычислительными возможностями.

Основное назначение больших ЭВМ — выполнение работ, связанных с обработкой и хранением больших объемов информации, проведением сложных расчетов и исследований в ходе решения вычислительных и информационно-логических задач. Такими машинами, как правило, оснащаются вычислительные центры, используемые совместно несколькими организациями. Большие машины составляли основу парка вычислительной техники до середины 70-х годов и успешно эксплуатируются поныне. К ним относятся большинство моделей фирмы IBM (семейства 360,370,390) и их отечественные аналоги ЕС ЭВМ.

В настоящее время высказываются полярные мнения о перспективах развития больших машин. Согласно одному из них, возможности больших машин полностью перекрываются, с одной стороны, супер-ЭВМ, а с другой — мини-ЭВМ и, выработав свой ресурс, этот класс прекратит свое существование. Другая сторона убеждает в необходимости развития универсальных больших и супер-ЭВМ, которые обладают способностью работать одновременно с большим количеством пользователей, создавать гигантские базы данных и обеспечивать эффективную вычислительную работу. К этому следует добавить, что большие ЭВМ обеспечивают устойчивость вычислительного процесса, безопасность информации и низкую стоимость ее обработки.

Производительность больших ЭВМ порой оказывается недостаточной для ряда приложений, например, таких как прогнозирование метеообстановки, ядерная энергетика, оборона и т. д. Эти обстоятельства стимулировали создание сверхбольших или суперЭВМ. Такие машины обладают колоссальным быстродействием в миллиарды операций в секунду, основанном на выполнении параллельных вычислений и использовании многоуровневой иерархической структуры ЗУ(запоминающих устройств), требуют для своего размещения специальных помещений и крайне сложны в эксплуатации. Стоимость отдельной ЭВМ такого класса достигает десятков миллионов долларов. Представители этого класса ЭВМ — компьютеры фирм Cray Research, Control Data Corporation (CDC) и отечественные супер-ЭВМ семейства Эльбрус.

Средние ЭВМ представляют некоторый интерес в историческом плане. На определенном этапе развития ЭВМ, когда их номенклатура и, соответственно, возможности были ограниченными, появление средних машин было закономерным. Вычислительные машины этого класса обладают несколько меньшими возможностями, чем большие ЭВМ, но зато им присуща и более низкая стоимость. Они предназначены для использования всюду, где приходится постоянно обрабатывать достаточно

большие объемы информации с приемлемыми временными затратами. В настоящее время трудно определить четкую грань между средними ЭВМ и большими с одной стороны и малыми — с другой. К средним могут быть отнесены некоторые модели ЕС ЭВМ, например: ЕС-1036, ЕС-1130, ЕС-1120. За рубежом средние ЭВМ выпускают фирмы IBM (International Business Machinary), DEC (Digital Equipment Corporation), Hewlett Packard, COMPAREX и др.

Малые ЭВМ составляют самый многочисленный и быстроразвивающийся класс ЭВМ. Их популярность объясняется малыми размерами, низкой стоимостью (по сравнению с большими и средними ЭВМ) и универсальными возможностями.

Класс **мини-ЭВМ** появился в 60-е годы (12-разрядная ЭВМ PD5-5 фирмы DEC). Их появление было обусловлено развитием элементной базы и избыточностью ресурсов больших и средних ЭВМ для ряда приложений. Для мини-ЭВМ характерно представление данных с узким диапазоном значений (машинное слово — 2 байта), использование принципа магистральности в архитектуре и более простое взаимодействие человека и ЭВМ. Такие машины широко применяются для управления сложными видами оборудования, создания систем автоматизированного проектирования и гибких производственных систем. К мини-ЭВМ относятся машины серии PDP (затем VAX) фирмы DEC и их отечественные аналоги — модели семейства малых ЭВМ (СМ ЭВМ).

При переходе от схем с малой и средней степенями интеграции к интегральным микросхемам с большой и сверхбольшой степенями интеграции оказалось возможным создание на одной БИС или СБИС функционально законченного устройства обработки информации, выполняющего функции процессора. Такое устройство принято называть ***микропроцессором.*** Изобретение микропроцессора привело к появлению еще одного класса ЭВМ — **микро-ЭВМ.** Определяющим признаком микро-ЭВМ является наличие одного или нескольких микропроцессоров. Создание микропроцессора не только изменило центральную часть ЭВМ, но и привело к необходимости разработки малогабаритных устройств ее периферийной части. Микро-ЭВМ, благодаря малым размерам, высокой производительности, повышенной надежности и небольшой стоимости нашли широкое распространение во всех сферах народного хозяйства и оборонного комплекса. С появлением микропроцессоров и микро-ЭВМ становится возможным создание так называемых ***интеллектуальных терминалов,*** выполняющих сложные процедуры предварительной обработки информации.

Успехи в развитии микропроцессоров и микро-ЭВМ привели к появлению **персональных ЭВМ** (ПЭВМ), предназначенных для индивидуального обслуживания пользователя и ориентированных на решение различных задач неспециалистами в области вычислительной техники. Все оборудование персональной ЭВМ размещается в пределах стола.

ПЭВМ, выпускаемые в сотнях тысяч и миллионах экземпляров, вносят коренные изменения в формы использования вычислительных средств, в значительной степени расширяют масштабы их применения. Они широко используются как для поддержки различных видов профессиональной деятельности (инженерной, административной, производственной, литературной, финансовой и др.), так и в быту, например для обучения и досуга.

Персональный компьютер позволяет эффективно выполнять научно-технические и финансово-экономические расчеты, организовывать базы данных, подготавливать и редактировать документы и любые другие тексты, вести делопроизводство, обрабатывать графическую информацию и т. д. Выполнение многих из указанных функций поддерживается многочисленными эффективными универсальными функциональными пакетами программ.

На основе ПЭВМ создаются ***автоматизированные рабочие места (АРМ)*** для представителей разных профессий (конструкторов, технологов, административного аппарата и др.).

Рынок персональных и микро-ЭВМ непрерывно расширяется за счет поставок ведущих мировых фирм: IBM, Compaq, Hewlett Packard, Apple (США), Siemens (Германия), ICL (Англия) и др.

БАЗОВАЯ АППАРАТНАЯ КОНФИГУРАЦИЯ

Персональный компьютер – универсальная техническая система. Его *конфигурацию* (состав оборудования) можно гибко изменять по мере необходимости. Тем не менее, существует понятие ***базовой конфигурации***, которую считают типовой. В таком комплекте компьютер обычно поставляется. Понятие базовой конфигурации может меняться. В настоящее время в базовой конфигурации рассматривают четыре устройства (рис. 3):

- системный блок;
- монитор;
- клавиатуру;
- мышь.

Помимо компьютеров с базовой конфигурации всё большее распространение получают мультимедийные компьютеры, оснащенные устройством чтения компакт-дисков, колонками и микрофоном.

Рис. 3. Конфигурация мультимедийного компьютера

Системный блок

Системный блок представляет собой основной узел, внутри которого

установлены наиболее важные компоненты. Устройства, находящиеся внутри системного блока, называют ***внутренними,*** а устройства, подключаемые к нему снаружи, называют ***внешними***. Внешние дополнительные устройства, предназначенные для ввода, вывода и длительного хранения данных, также называют ***периферийными***.

По внешнему виду системные блоки различаются формой корпуса. Выбор того или иного типа корпуса определяется вкусом и потребностями модернизации компьютера. Наиболее оптимальным типом корпуса для большинства пользователей является корпус типа *mini tower*.

Кроме формы, для корпуса важен параметр, называемый *форм-фактором.* От него зависят требования к размещаемым устройствам. В настоящее время в основном используются корпуса двух форм-факторов: *AT и ATX* . Форм-фактор корпуса должен быть обязательно согласован с форм-фактором главной (системной) платы компьютера, так называемой *материнской платы*.

Материнская плата

Материнская плата (mother board) – основная плата персонального компьютера, представляющая из себя лист стеклотекстолита, покрытый медной фольгой. Путем травления фольги получают тонкие медные проводники соединяющие электронные компоненты. На материнской плате размещаются:

- ***процессор*** – основная микросхема, выполняющая большинство математических и логических операций;
- ***шины*** – наборы проводников, по которым происходит обмен сигналами между внутренними устройствами компьютера;
- ***оперативная память (оперативное запоминающее устройство, ОЗУ)*** – набор микросхем, предназначенных для временного хранения данных, когда компьютер включен;
- ***ПЗУ (постоянное запоминающее устройство)*** – микросхема, предназначенная для длительного хранения данных, в том числе и когда компьютер выключен;
- ***микропроцессорный комплект (чипсет)*** – набор микросхем, управляющих работой внутренних устройств компьютера и определяющих основные функциональные возможности материнской платы;
- разъемы для подключения дополнительных устройств (слоты).

Менялись микропроцессоры, рождались и умирали системные и локальные шины, а вид и размеры материнской платы практически не менялись с 1984 г. Например, размер оригинальной материнской платы IBM PC/AT под названием Baby-AT был равен 217 на 331 мм, а размеры современной материнской платы P3B-F равны 192 мм на 304 мм. (рис. 4).

Некоторые из них имеют отличительный признак – наличие гнезда для подключения микропроцессора с нулевым усилием сопряжения – ZIF (Zero Insertion Force) типа Socket 7. В данное гнездо можно ставить как процессор Intel Pentium, так и процессоры Cyrix 6x86, AMD K5 и K6. Более современные платы имеют разъем Slot 1 или Slot 2 для подключения

процессоров Pentium II и Pentium III.

Процессор

Процессор (микропроцессор, центральный процессор, CPU) – основная микросхема компьютера, в которой и производятся все вычисления. Он представляет из себя большую микросхему (например, размеры микропроцессора Pentium примерно 5*5*0,5 см), которую можно легко найти на материнской плате. На процессоре установлен большой медный ребристый радиатор, охлаждаемый вентилятором. Конструктивно процессор состоит из ячеек, в которых данные могут не только храниться, но и изменяться. Внутренние ячейки процессора называют ***регистрами***. Важно также отметить, что данные, попавшие в некоторые регистры, рассматриваются не как данные, а как команды, управляющие обработкой данных в других регистрах. Среди регистров процессора есть и такие, которые в зависимости от своего содержания способны модифицировать исполнение команд. Таким образом, управляя засылкой данных в разные регистры процессора, можно управлять обработкой данных. На этом и основано исполнение программ.

С остальными устройствами компьютера, и в первую очередь с оперативной памятью, процессор связан несколькими группами проводников, называемых *шинами.* Основных шин три: *шина данных, адресная тина* и *командная шина.*

Адресная шина. У процессоров Intel Pentium (а именно они наиболее распространены в персональных компьютерах) адресная шина 32-разрядная, то есть состоит из 32 параллельных линий. В зависимости от того, есть напряжение на какой-то из линий или нет, говорят, что на этой линии выставлена единица или ноль. Комбинация из 32 нулей и единиц образует 32-разрядный адрес, указывающий на одну из ячеек оперативной памяти. К ней и подключается процессор для копирования данных из ячейки в один из своих регистров.

Шина данных. По этой шине происходит копирование данных из оперативной памяти в регистры процессора и обратно. В компьютерах, собранных на базе процессоров Intel Pentium, шина данных 64-разрядная, то есть состоит из 64 линий, по которым за один раз на обработку поступают сразу 8 байтов.

Шина команд. Для того чтобы процессор мог обрабатывать данные, ему нужны команды. Он должен знать, что следует сделать с теми байтами, которые хранятся в его регистрах. Эти команды поступают в процессор тоже из оперативной памяти, но не из тех областей, где хранятся массивы данных, а оттуда, где хранятся программы. Команды тоже представлены в виде байтов. Самые простые команды укладываются в один байт, однако есть и такие, для которых нужно два, три и более байтов. В большинстве современных процессоров шина команд 32-разрядная (например, в процессоре Intel Pentium), хотя существуют 64-разрядные процессоры и даже 128-разрядные.

Система команд процессора. В процессе работы процессор

обслуживает данные, находящиеся в его регистрах, в поле оперативной памяти, а также данные, находящиеся во внешних портах процессора. Часть данных он интерпретирует непосредственно как данные, часть данных – как адресные данные, а часть – как команды. Совокупность всех возможных команд, которые может выполнить процессор над данными, образует так называемую *систему команд процессора*. Процессоры, относящиеся к одному семейству, имеют одинаковые или близкие системы команд. Процессоры, относящиеся к разным семействам, различаются по системе команд и невзаимозаменяемыми.

Совместимость процессоров. Если два процессора имеют одинаковую систему команд, то они полностью совместимы на программном уровне. Это означает, что программа, написанная для одного процессора, может исполняться и другим процессором. Процессоры, имеющие разные системы команд, как правило, несовместимы или ограниченно совместимы на программном уровне.

Группы процессоров, имеющих ограниченную совместимость, рассматривают как ***семейства процессоров***. Так, например, все процессоры Intel Pentium относятся к так называемому семейству x86. Родоначальником этого семейства был 16-разрядный процессор Intel 8086, на базе которого собиралась первая модель компьютера *IBM PC*. Впоследствии выпускались процессоры Intel 80286, Intel 80386, Intel 80486, Intel Pentium 60,66,75,90,100,133; несколько моделей процессоров Intel Pentium MMX, модели Intel Pentium Pro, Intel Pentium II, Intel Celeron, Intel Xeon, Intel Pentium III (см. рис. 5,а), Intel Pentium IV и другие. Все эти модели, и не только они, а также многие модели процессоров компаний AMD (см. рис. 5,б) и Cyrix относятся к семейству x86 и обладают совместимостью по принципу «сверху вниз».

Основные параметры процессоров. Основными параметрами процессоров являются: *рабочее напряжение, разрядность, рабочая тактовая частота, коэффициент внутреннего умножения тактовой частоты* и размер *кэш-памяти.*

Рабочее напряжение процессора обеспечивает материнская плата, поэтому разным маркам процессоров соответствуют разные материнские платы (их надо выбирать совместно). По мере развития процессорной техники происходит постепенное понижение рабочего напряжения. Ранние модели процессоров x86 имели рабочее напряжение 5 В. С переходом к процессорам Intel Pentium оно было понижено до 3,3 В, а в настоящее время оно составляет менее 3 В. Причем ядро процессора питается пониженным напряжением 2,2 В. Понижение рабочего напряжения позволяет уменьшить расстояния между структурными элементами в кристалле процессора до десятитысячных долей миллиметра, не опасаясь электрического пробоя. Пропорционально квадрату напряжения уменьшается и тепловыделение в процессоре, а это позволяет увеличивать его производительность без угрозы перегрева.

Разрядность процессора показывает, сколько бит данных он может

принять и обработать в своих регистрах за один раз *(за один такт)*. Первые процессоры x86 были 16-разрядными. Начиная с процессора 80386 они имеют 32-разрядную архитектуру. Современные процессоры семейства Intel Pentium остаются 32-разрядными, хотя и работают с 64-разрядной шиной данных (разрядность процессора определяется не разрядностью шины данных, а разрядностью командной шины).

Тактовые сигналы процессор получает от материнской платы, которая, в отличие от процессора, представляет собой не кристалл кремния, а большой набор проводников и микросхем. По чисто физическим причинам материнская плата не может работать со столь высокими частотами, как процессор. Сегодня ее предел составляет 100-133 МГц. Для получения более высоких частот в процессоре происходит ***внутреннее умножение частоты*** на коэффициент 3; 3,5; 4; 4,5; 5 и более.

Обмен данными внутри процессора происходит в несколько раз быстрее, чем обмен с другими устройствами, например с оперативной памятью. Для того чтобы уменьшить количество обращений к оперативной памяти, внутри процессора создают буферную область – так называемую ***кэш-память***. Это как бы «сверхоперативная память». Когда процессору нужны данные, он сначала обращается в кэш-память, и только если там нужных данных нет, происходит его обращение в оперативную память. Принимая блок данных из оперативной памяти, процессор заносит его одновременно и в кэш-память. «Удачные» обращения в кэш-память называют *попаданиями в кэш.* Процент попаданий тем выше, чем больше размер кэш-памяти, поэтому высокопроизводительные процессоры комплектуют повышенным объемом кэш-памяти.

Кэш-память третьего уровня выполняют на быстродействующих микросхемах типа *SRAM* и размещают на материнской плате вблизи процессора. Ее объемы могут достигать нескольких Мбайт, но работает она на частоте материнской платы.

Шинные интерфейсы материнской платы

Связь между всеми собственными и подключаемыми устройствами материнской платы выполняют ее шины и логические устройства, размещенные в микросхемах микропроцессорного комплекта (чипсета). От архитектуры этих элементов во многом зависит производительность компьютера.

ISA. Историческим достижением компьютеров платформы *IBM PC* стало внедрение почти двадцать лет назад архитектуры, получившей статус *промышленного стандарта ISA (Industry Standard Architecture).* Она не только позволила связать все устройства системного блока между собой, но и обеспечила простое подключение новых устройств через стандартные разъемы (слоты). Пропускная способность шины, выполненной по такой архитектуре, составляет до 5,5 Мбайт/с, но, несмотря на низкую пропускную способность, эта шина продолжает использоваться в компьютерах для подключения сравнительно «медленных» внешних устройств, например звуковых карт и модемов.

EISA. Расширением стандарта *ISA* стал стандарт *EISA (Extended ISA),* отличающийся увеличенным разъемом и увеличенной производительностью (до 32 Мбайт/с). Как и *ISA, в* настоящее время данный стандарт считается устаревшим. После 2000 года выпуск материнских плат с разъемами *ISA/EISA* и устройств, подключаемых к ним, прекращается.

VLB. Название интерфейса переводится как *локальная шина стандарта VESA (VESA Local Bus).* Понятие «локальной шины» впервые появилось в конце 80-х годов. Оно связано тем, что при внедрении процессоров третьего и четвертого поколений (Intel 80386 и Intel 80486) частоты основной шины (в качестве основной использовалась шина *ISA/EISA)* стало недостаточно для обмена между процессором и . оперативной памятью. Локальная шина, имеющая повышенную частоту, связала между собой процессор и память в обход основной шины. Впоследствии в эту шину «врезали» интерфейс для подключения видеоадаптера, который тоже требует повышенной пропускной способности, – так появился стандарт *VLB,* который позволил поднять тактовую частоту локальной шины до 50 МГц и обеспечил пиковую пропускную способность до 130 Мбайт/с.

Основным недостатком интерфейса *VLB* стало то, что предельная частота локальной шины и, соответственно, ее пропускная способность зависят от числа устройств, подключенных к шине. Так, например, при частоте 50 Мгц к шине может быть подключено только одно устройство (видеокарта). Для сравнения скажем, что при частоте 40 Мгц возможно подключение двух, а при частоте 33 МГц – трех устройств.

PCI. Интерфейс *PCI (Peripheral Component Interconnect – стандарт подключения, внешних компонентов)* был введен в персональных компьютерах, выполненных на базе процессоров Intel Pentium. По своей сути это тоже интерфейс локальной шины, связывающей процессор с оперативной памятью, в которую врезаны разъемы для р. подключения внешних устройств. Для связи с основной шиной компьютера *(ISA/EISA)* используются специальные интерфейсные преобразователи – *мосты PCI (PCI Bridge).* В современных компьютерах функции моста *PCI* выполняют микросхемы микропроцессорного комплекта (чипсета).

FSB. Шина *PCI,* появившаяся в компьютерах на базе процессоров Intel Pentium как локальная шина, предназначенная для связи процессора с оперативной памятью, недолго оставалась в этом качестве. Сегодня она используется только как шина для подключения внешних устройств, а для связи процессора и памяти, начиная с процессора Intel Pentium Pro используется специальная шина, получившая название *Front Side Bus (FSB)*. Эта шина работает на очень высокой частоте 100-125 МГц. В настоящее время внедряются материнские платы с частотой шины *FSB* 133 МГц и ведутся разработки плат с частотой до 200 МГц. Частота шины *FSB* является одним из основных потребительских параметров – именно он и указывается в спецификации материнской платы. Пропускная способность шины *FSB* при частоте 100 МГц составляет порядка 800 Мбайт/с.

AGP. Видеоадаптер – устройство, требующее особенно высокой

скорости передачи данных. Как при внедрении локальной шины *VLB,* так и при внедрении локальной шины *PCI* видеоадаптер всегда был первым устройством, «врезаемым» в новую шину. Сегодня параметры шины *PCI* уже не соответствуют требованиям видеоадаптеров, поэтому для них разработана отдельная шина, получившая название *AGP (Advanced Graphic Port – усовершенствованный графический порт).* Частота этой шины соответствует частоте шины *PCI* (33 МГц или 66 МГц), но она имеет много более высокую пропускную способность – до 1066 Мбайт/с (в режиме четырехкратного умножения).

PCMCIA *(Personal Computer Memory Card International Association – стандарт международной ассоциации производителей плат памяти для персональных компьютеров).* Этот стандарт определяет интерфейс подключения плоских карт памяти небольших размеров и используется в портативных персональных компьютерах.

USB *(Universal Serial Bus – универсальная последовательная магистраль).* Это одно из последних нововведений в архитектурах материнских плат. Этот стандарт определяет способ взаимодействия компьютера с периферийным оборудованием. Он позволяет подключать до 256 различных устройств, имеющих последовательный интерфейс. Устройства могут включаться цепочками (каждое следующее устройство подключается к предыдущему). Производительность шины *USB* относительно невелика и составляет до 1,5 Мбит/с, но для таких устройств, как клавиатура, мышь, модем, джойстик и т. п., этого достаточно. Удобство шины состоит в том, что она практически исключает конфликты между различным оборудованием, позволяет подключать и отключать устройства в «горячем режиме» (не выключая компьютер) и позволяет объединять несколько компьютеров в простейшую локальную сеть без применения специального оборудования и программного обеспечения.

Параметры микропроцессорного комплекта (чипсета) в наибольшей степени определяют свойства и функции материнской платы. В настоящее время большинство чипсетов материнских плат выпускаются на базе двух микросхем, получивших название «северный мост» и «южный мост».

«Северный мост» управляет взаимосвязью четырех устройств: процессора, оперативной памяти, порта *AGP* и шины *PCI.* Поэтому его также называют ***четырехпортовым контроллером***.

«Южный мост» называют также ***функциональным контроллером***. Он выполняет функции контроллера жестких и гибких дисков, функции моста *ISA – PCI,* контроллера клавиатуры, мыши, шины *USB* и т. п.

Оперативная память

Оперативная память (RAM – Random Access Memory) – это массив кристаллических ячеек, способных хранить данные. Существует много различных типов оперативной памяти, но с точки зрения физического принципа действия различают *динамическую память (DRAM)* и *статическую память (SRAM).*

Ячейки динамической памяти (DRAM) можно представить в виде

микроконденсаторов, способных накапливать заряд на своих обкладках. Это наиболее распространенный и экономически доступный тип памяти. Недостатки этого типа связаны, во-первых, с тем, что как при заряде, так и при разряде конденсаторов неизбежны переходные процессы, то есть запись данных происходит сравнительно медленно. Второй важный недостаток связан с тем, что заряды ячеек имеют свойство рассеиваться в пространстве, причем весьма быстро. Если оперативную память постоянно не «подзаряжать», утрата данных происходит через несколько сотых долей секунды. Для борьбы с этим явлением в компьютере происходит постоянная ***регенерация (освежение, подзарядка)*** ячеек оперативной памяти. Регенерация осуществляется несколько десятков раз в секунду и вызывает непроизводительный расход ресурсов вычислительной системы.

Одна адресуемая ячейка содержит восемь двоичных ячеек, в которых можно сохранить 8 бит, то есть один байт данных. Таким образом, адрес любой ячейки памяти можно выразить четырьмя байтами.

Оперативная память в компьютере размещается на стандартных панельках, называемых ***модулями***. Модули оперативной памяти вставляют в соответствующие разъемы на материнской плате. Если к разъемам есть удобный доступ, то операцию можно выполнять своими руками. Если удобного доступа нет, может потребоваться неполная разборка узлов системного блока, и в таких случаях операцию поручают специалистам.

Конструктивно модули памяти имеют два исполнения – ***однорядные*** *(SIMM-модули)* и ***двухрядные*** *(DIMM-модули)*. На компьютерах с процессорами Pentium однорядные модули можно применять только парами (количество разъемов для их установки на материнской плате всегда четное), а *DIMM-модули* можно устанавливать по одному. Многие модели материнских плат имеют разъемы как того, так и другого типа, но комбинировать на одной плате модули разных типов нельзя.

Основными характеристиками модулей оперативной памяти являются объем памяти и время доступа. *SIMM*-модули поставляются объемами 4,8,16,32 Мбайт, а *DIMM*-модули – 16, 32, 64, 128 Мбайт и более. Время доступа показывает, сколько времени необходимо для обращения к ячейкам памяти – чем оно меньше, тем лучше. Время доступа измеряется в миллиардных долях секунды *(наносекундах,* нс). Типичное время доступа к оперативной памяти для *SIMM*-модулей – 50-70 нс. Для современных *DIMM*-модулей оно составляет 7-10 нс.

Жесткий диск

Жесткий диск – основное устройство для долговременного хранения больших объемов данных и программ. На самом деле это не один диск, а группа соосных дисков, имеющих магнитное покрытие и вращающихся с высокой скоростью. Таким образом, этот «диск» имеет не две поверхности, как должно быть у обычного плоского диска, а 2n поверхностей, где *n* – число отдельных дисков в группе.

Над каждой поверхностью располагается головка, предназначенная для чтения-записи данных. При высоких скоростях вращения дисков (90 об/с) в

зазоре между головкой и поверхностью образуется аэродинамическая подушка, и головка парит над магнитной поверхностью на высоте, составляющей несколько тысячных долей миллиметра. При изменении силы тока, протекающего через головку, происходит изменение напряженности динамического магнитного поля в зазоре, что вызывает изменения в стационарном магнитном поле ферромагнитных частиц, образующих покрытие диска. Так осуществляется запись данных на магнитный диск.

Дисковод гибких дисков

Информация на жестком диске может храниться годами, однако иногда требуется ее перенос с одного компьютера на другой. Несмотря на свое название, жесткий диск является весьма хрупким прибором, чувствительным к перегрузкам, ударам и толчкам. Теоретически, переносить информацию с одного рабочего места на другое путем переноса жесткого диска возможно, и в некоторых случаях так и поступают, но все-таки этот прием считается нетехнологичным, поскольку требует особой аккуратности и определенной квалификации.

Для оперативного переноса небольших объемов информации используют так называемые ***гибкие магнитные диски (дискеты)***, которые вставляют в специальный накопитель – ***дисковод***. Приемное отверстие накопителя находится на лицевой панели системного блока. Правильное направление подачи гибкого диска отмечено стрелкой на его пластиковом кожухе.

Основными параметрами гибких дисков являются: технологический размер (измеряется в дюймах), плотность записи (измеряется в кратных единицах) и полная емкость.

Гибкие диски размером 3,5 дюйма выпускают с 1980 года. Односторонний диск ***обычной плотности*** имел емкость 180 Кбайт, двусторонний – 360 Кбайт, а ***двусторонний двойной плотности*** – 720 Кбайт. Ныне стандартными считают диски размером 3,5 дюйма ***высокой плотности***. Они имеют емкость **1440 Кбайт (1,4 Мбайт)** и маркируются буквами ***HD (high density – высокая плотность).***

Дисковод компакт-дисков CD-ROM

В период 1994-1995 годах в базовую конфигурацию персональных компьютеров перестали включать дисководы гибких дисков диаметром 5,25 дюйма, но вместо них стандартной стала считаться установка дисковода *CD-ROM,* имеющего такие же внешние размеры.

Аббревиатура ***CD-ROM (Compact Disc Read-Only Memory)*** переводится на русский язык как ***постоянное запоминающее устройство на основе компакт-диска***. Принцип действия этого устройства состоит в считывании числовых данных с помощью лазерного луча, отражающегося от поверхности диска. Цифровая запись на компакт-диске отличается от записи на магнитных дисках очень высокой плотностью, и стандартный компакт-диск может хранить примерно 650 Мбайт данных.

Большие объемы данных характерны для ***мультимедийной информации*** (графика, музыка, видео), поэтому дисководы *CD-ROM* относят

к аппаратным средствам мультимедиа. Программные продукты, распространяемые на лазерных дисках, называют *мультимедийными изданиями.* Сегодня мультимедийные издания завоевывают все более прочное место среди других традиционных видов изданий. Так, например, существуют книги, альбомы, энциклопедии и даже периодические издания (электронные журналы), выпускаемые на *CD-ROM.*

Монитор

Монитор – устройство визуального представления данных. Это не единственно возможное, но главное устройство вывода. Его основными потребительскими параметрами являются: размер и шаг маски экрана, максимальная частота регенерации изображения, класс защиты.

По способу формирования изображения мониторы делятся на ***жидкокристаллические (LCD)*** и построенные на основе ***электронно-лучевой трубки (CRT)***.

Клавиатура

Клавиатура – клавишное устройство управления персональным компьютером. Служит для ввода *алфавитно-цифровых (знаковых)* данных, а также команд управления. Комбинация монитора и клавиатуры обеспечивает простейший *интерфейс пользователя. С* помощью клавиатуры управляют компьютерной системой, а с помощью монитора получают от нее отклик.

Известно два типа клавиатур – *клавиатура с* ***механическими*** *и* ***мембранными переключателями***. Переключатель первого типа это обычный механический датчик, традиционное устройство известное уже несколько десятилетий. Второй тип датчика устроен несколько сложнее. Переключатель данного типа представляет из себя набор мембран, при нажатии на клавишу верхняя мембрана прогибается и через специальное отверстие в изолирующей мембране замыкается на нижнюю мембрану. Как правило, предпочтение отдается клавиатуре с механическими датчиками. Они выдерживают многолетнюю эксплуатацию, надежны и поддаются ремонту в случае необходимости.

Из данного объяснения ясно, что каждой клавише присвоен уникальный цифровой код и существуют специальные таблицы кодировки клавиатуры. Например, кодовая таблица США имеет номер 437 (как правило, она записана в специальную микросхему – знакогенератор процессора), а кодовая страница России имеет номер 866. Для смены кодировки клавиатуры применяются специальные программы – клавиатурные драйверы. Современные клавиатуры способны не только передавать данные в процессор, но и воспринимать команды от него.

Для разных языков существуют различные схемы закрепления символов национальных алфавитов за конкретными алфавитно-цифровыми клавишами. Такие схемы называются ***раскладками клавиатуры***. Переключения между различными раскладками выполняются программным образом – это одна из функций операционной системы. Соответственно, способ переключения зависит от того, в какой операционной системе работает компьютер. Например, в системе Windows 98 для этой цели могут

использоваться следующие комбинации: левая клавиша ALT+SHIFT или CTRL+SHIFT. При работе с другой операционной системой способ переключения можно установить по справочной системе той программы, которая выполняет переключение.

Мышь

Мышь – устройство управления манипуляторного типа. Представляет собой плоскую коробочку с двумя-тремя кнопками. Перемещение мыши по плоской поверхности синхронизировано с перемещением графического объекта *(указателя мыши)* на экране монитора.

Принцип действия. В отличие от рассмотренной ранее клавиатуры, мышь не является стандартным органом управления, и персональный компьютер не имеет для нее выделенного порта. Для мыши нет и постоянного выделенного прерывания, а базовые средства ввода и вывода *(BIOS)* компьютера, размещенные в постоянном запоминающем устройстве (ПЗУ), не содержат программных средств для обработки прерываний мыши.

ТЕМА: **Виды программного обеспечения ЭВМ**

Назначением ЭВМ является выполнение программ. Программа содержит команды, определяющие порядок действий компьютера. Совокупность программ для компьютера образует ***программное обеспечение (ПО).*** По функциональному признаку различают следующие виды ПО:

- системное;
- прикладное.

Под ***системным (базовым)*** понимается программное обеспечение, включающее в себя операционные системы, сетевое ПО, сервисные программы, а также средства разработки программ (трансляторы, редакторы связей, отладчики и пр.).

Основные функции ***операционных систем*** (ОС) заключаются в управлении ресурсами (физическими и логическими) и процессами вычислительных систем. Физическими ресурсами являются: оперативная память, процессор, монитор, печатающее устройство, магнитные и оптические диски. К логическим ресурсам можно отнести программы, файлы, события и т. д. Под процессом понимается некоторая последовательность действий, предписанная соответствующей программой и используемыми ею данными.

В настоящее время существует большое количество ОС, разработанных для ЭВМ различных типов. На ЭВМ Единой Системы (ЕС ЭВМ), например, использовались такие операционные системы, как СВМ и ОС ЕС, на малых ЭВМ (СМ-4, СМ-1420 и др.) - ОС РВ и RSX-11. На персональных ЭВМ долгое время эксплуатировалась ОС-MS-DOS. В настоящее время получили распространение системы Windows 98/Me, Windows 2000, Linix.

Сетевое ПО предназначено для управления общими ресурсами в распределенных вычислительных системах: сетевыми накопителями на магнитных дисках, принтерами, сканерами, передаваемыми сообщениями и т. д. К сетевому ПО относят ОС, поддерживающие работу ЭВМ в сетевых

конфигурациях (так называемые сетевые ОС), а также отдельные сетевые программы (пакеты), используемые совместно с обычными, не сетевыми ОС.

Например, большое распространение получили следующие сетевые ОС: NetWare 4.1 (фирма Novell), Windows NT Server 3.5 (фирма Microsoft) и LAN Server 4.0 Advanced (фирма IBM). Однако в последнее время лидирующие позиции начинает занимать ОС Windows 2000 Server фирмы Microsoft.

Для расширения возможностей операционных систем и предоставления набора дополнительных услуг используются ***сервисные программы.*** Их можно разделить на следующие группы:

- интерфейсные системы;
- оболочки операционных систем;
- утилиты.

Интерфейсные системы являются естественным продолжением операционной системы и модифицируют как пользовательский, так и программный интерфейсы, а также реализуют дополнительные возможности по управлению ресурсами ЭВМ. В связи с тем, что развитая интерфейсная система может изменить весь пользовательский интерфейс, часто их также называют операционными системами. Это относится, например, к Windows 3.11 и Windows 3.11 for WorkGroups (для рабочих групп).

Оболочки операционных систем, в отличие от интерфейсных систем, модифицируют только пользовательский интерфейс, предоставляя пользователю качественно новый интерфейс по сравнению с реализуемым операционной системой. Такие системы существенно упрощают выполнение часто запрашиваемых функций, например, таких операций с файлами, как копирование, переименование и уничтожение, а также предлагают пользователю ряд дополнительных услуг. В целом, программы-оболочки заметно повышают уровень пользовательского интерфейса, наиболее полно удовлетворяя потребностям пользователя.

На ПЭВМ широко используются такие программы-оболочки, как Norton Commander, FAR Manager и Windows Commander.

Утилиты предоставляют пользователям средства обслуживания компьютера и его ПО. Они обеспечивают реализацию следующих действий:

- обслуживание магнитных дисков;
- обслуживание файлов и каталогов;
- предоставление информации о ресурсах компьютера;
- шифрование информации;
- защита от компьютерных вирусов;
- архивация файлов и др.

Существуют ***отдельные утилиты,*** используемые для решения одного из перечисленных действий, и ***многофункциональные комплекты утилит***. В настоящее время для ПЭВМ среди многофункциональных утилит одним из наиболее совершенных является комплект утилит Norton Utilities. Существуют его версии для использования в среде DOS и Windows.

Средства разработки программ используются для разработки нового программного обеспечения как системного, так и прикладного.

Прикладным называется ПО, предназначенное для решения определенной целевой задачи из проблемной области. Часто такие программы называют *приложениями.*

Спектр проблемных областей в настоящее время весьма широк и включает в себя по крайней мере следующие: промышленное производство, инженерную практику, научные исследования, медицину, управление (менеджмент), делопроизводство, издательскую деятельность, образование и т. д.

Из всего разнообразия прикладного ПО выделяют группу наиболее распространенных программ (типовые пакеты и программы), которые можно использовать во многих областях человеческой деятельности.

К *типовому прикладному ПО* относят следующие программы:

- текстовые процессоры;
- табличные процессоры;
- системы иллюстративной и деловой графики (графические процессоры);
- системы управления базами данных;
- экспертные системы;
- программы математических расчетов, моделирования и анализа экспериментальных данных.

Предлагаемые на рынке ПО приложения, в общем случае, могут быть выполнены как отдельные программы либо как интегрированные системы. Интегрированными системами обычно являются экспертные системы, программы математических расчетов, моделирования и анализа экспериментальных данных, а также офисные системы. Примером мощной и широко распространенной интегрированной системы является офисная система Microsoft Office.

Поскольку разработка ПО любого назначения, как правило, является довольно сложным и трудоемким процессом, дальнейший материал настоящего раздела посвятим общим вопросам разработки программ и инструментальному ПО.

Операционная система MS-DOS

Операционная система MS-DOS – это однопользовательская, однозадачная, не сетевая 16-разрядная операционная система (ОС), ориентированная на использование на ПЭВМ с микропроцессором Intel 8088(80286).

Краткая история появления данной операционной системы такова. В октябре 1980 г. менеджеры фирмы IBM занялись поисками ОС для своего 16-разрядного персонального компьютера (ПК), находящегося в стадии разработки. В тот период на ПЭВМ наиболее широко применялась ОС CP/M (Control Program for MicroComputers) фирмы Digital Research. Не достигнув приемлемых соглашений с Digital Research, фирма IBM обратилась к фирме Microsoft (президент – Билл Гейтс). В тот момент у Microsoft не было соответствующей ОС, но ей была известна небольшая фирма Seattle Computer Products, которая имела такую ОС. За 50 000$ Билл Гейтс приобрел

права на эту ОС. В дальнейшем эта ОС послужила основой для MS-DOS. В ноябре 1980 г. Microsoft и IBM подписали договор на разработку ОС для IBM PC. В феврале 1981 г. появилась первая версия PC/MS-DOS, которая работала на IBM PC, в августе того же года – PC DOS 1.0 (эта версия была утверждена для применения на IBM PC).

Операционная система MS-DOS позволяет полностью использовать возможности процессоров Intel 8088 и Intel 80286, работающих в реальном режиме.

Основными характеристиками данной ОС являются:

– максимальный объем адресуемой физической памяти – 640 Кбайт;

– максимальный объем памяти, доступный из прикладных программ 640 Кбайт. Последние версии MS-DOS (начиная с 5.0) могут использовать адресное пространство между 640 Кбайт и 1 Мбайт для размещения своих составных частей и некоторых драйверов, освобождая тем самым память в адресном пространстве 0-640 Кбайт для использования прикладными программами;

– представление всех ресурсов персонального компьютера для одной, активной в настоящий момент, программы;

– развитая файловая система и процессор командного языка;

– слабая поддержка интерактивных средств взаимодействия с пользователем;

– занимаемый объем на диске, в зависимости от версии, от 1 Мбайта до 6 Мбайт. (минимум, при котором можно работать – 100 Кбайт).

Основные составные части MS-DOS

MS-DOS состоит из следующих компонент:

– блок начальной загрузки (размещается в 1-м секторе 0-дорожки 0-стороны системной дискеты);

– модуль расширения BIOS (IO.SYS для версии 5.0 и выше);

– модуль обработки прерываний (MSDOS.SYS для версии 5.0 и выше),

– командный процессор (COMMAND.COM);

– внешние команды (программы) MS-DOS;

– драйверы устройств;

– файл Config.SYS;

– файл Autoexec.bat.

Ядро MS-DOS включает блок начальной загрузки и файлы IO.SYS, MSDOS.SYS.

Блок начальной загрузки размещается в 1-м секторе 0-дорожки 0-стороны системной дискеты и/или в 1-м секторе HDD-диска, в разделе, отведенном под DOS. Выполняет следующие функции: просматривает корневой каталог системного диска и проверяет, являются ли **первые два файла** в каталоге – файлами **IO.SYS** и **MSDOS.SYS**. Если *ДА* – загружает их в ОЗУ и передает управление MS-DOS, если *НЕТ* – выдает сообщение на экране и ожидает нажатия какой-либо клавиши пользователем:

Non-System disk or disk error

Replace and press any key when ready

Не системный диск или ошибка диска
Замените и нажмите какую-либо клавишу, когда будет готово

Именно поэтому , при создании *системной дискеты* необходимо переносить на неё файлы *IO.SYS* и *MSDOS.SYS* с помощью специальной программы **SYS.COM**.

Модуль расширения BIOS IO.SYS

Это *резидентный модуль* (всегда находится в ОЗУ после загрузки, пока включен ПК). Взаимодействует с *BIOS.* Расширяет возможности BIOS или изменяет ее свойства (там, где необходимо) с помощью дополнительных драйверов.

Модуль обработки прерываний MSDOS.SYS

Это *резидентный модуль.* Обеспечивает интерфейс высокого уровня для прикладных программ, содержит программные средства для управления файлами, устройствами ввода-вывода, обработки исключительных ситуаций (ошибок) и др. Прикладная программа вызывает функции этого модуля через механизм программных прерываний, передавая (принимая) информацию к (от) MS-DOS через регистры CPU или (и) области памяти ОЗУ. MSDOS.SYS транслирует (переводит) запрос прикладной программы в один или несколько вызовов IO.SYS + BIOS.

Командный процессор COMMAND.COM

Отдельный модуль MS-DOS. Этот модуль может быть заменен на другой, более удобный. Предназначен для приема команд с клавиатуры или из *.bat - файлов и их выполнения; выполнения команд файла Autoexec.bat при загрузке MS-DOS; загрузки в ОЗУ и запуск на выполнение прикладных программ в среде MS-DOS.

Командный процессор состоит из 3-х частей :

– резидентной (размещается в ОЗУ сразу после MSDOS.SYS, включает процедуры обслуживания некоторых прерываний, процедуры обработки стандартных ошибок MS-DOS, процедуру загрузки транзитной части командного процессора);
– инициализирующей (в ОЗУ следует сразу за резидентной частью; во время загрузки ОС ей передается управление; она выполняет файл Autoexec.bat и некоторые другие действия; эта часть командного процессора стирается из ОЗУ первой же загруженной программой);
– транзитной (загружается в старшие адреса ОЗУ; обрабатывает все внутренние команды, команды с клавиатуры и из *.bat-файлов; выдает системную подсказку MS-DOS; загружает в ОЗУ программы и передает им управление).

Драйверы устройств

Специальные резидентные программы, которые управляют внешними устройствами. Драйверы загружаются в ОЗУ в том порядке, как они указаны в файле CONFIG.SYS.

Файл CONFIG.SYS

Специальный текстовый файл, где содержится информация о подгружаемых дополнительных драйверах и некоторая другая информация,

касающаяся непосредственно MS-DOS и выполняемых в ее среде прикладных программ. MS-DOS выполняет этот файл автоматически, сразу после загрузки COMMAND.COM.

Файл AUTOEXEC.BAT

Специальный текстовый файл, в котором содержится дополнительная настроечная информация. MS-DOS выполняет этот файл автоматически, сразу после выполнения файла CONFIG.SYS

***ТЕМА*: прикладное программное обеспечение**

Табличные процессоры, принципы их построения и обработки.

Microsoft Excel представляет много наглядных ориентиров, помогающих при работе с данными. Но Excel – чрезвычайно мощная программа, позволяющая достигать нужного результата различными способами даже в самых, казалось бы, простых ситуациях.

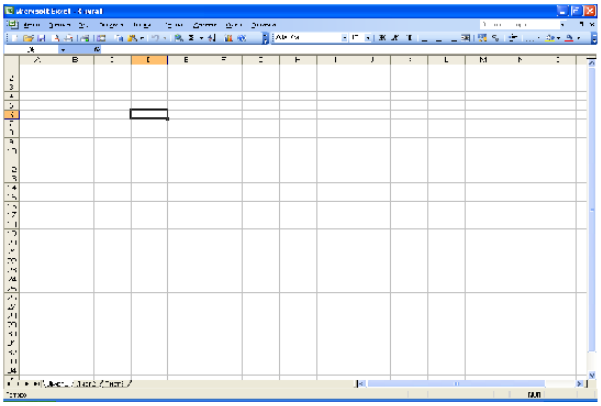


Рисунок 1 – Рабочий лист MS Excel

Ввод данных

В ячейки листа можно вводить два типа данных: константы и формулы. Константы разделяются на три основные категории: числовые значения, текстовые значения и значения дат и времени. Кроме того, в Excel имеются ещё два специальных типа констант: логические значения и ошибочные значения. Чтобы ввести числовое значение, выделите ячейку и введите с клавиатуры число. Вводимые цифры отображаются в строке формул и в активной ячейке. Ввод текста аналогичен вводу числовых значений. чтобы ввести значения в несколько смежных ячеек, сначала выделите эти ячейки. Затем используйте клавиши Enter, Shift+Enter, Tab и Shift+Tab для перемещения активной ячейки внутри диапазона в соответствии со следующей информацией

Клавиши	Активизируется
Enter	Ячейка ниже активной
Shift+Enter	Ячейка выше активной
Tab	Ячейка в соседнем столбце справа от активной
Shift+Tab	Ячейка в соседнем столбце слева от активной

Чтобы ввести текстовое значение, выделите ячейку, наберите на клавиатуре текст и зафиксируйте ввод, нажав клавишу Enter.

Скрытие ячеек и листов

К ячейкам листов можно применить режим скрытия формул, т.е. при активизации таких ячеек содержащихся в них формул не выводятся в строке формул. Сами формулы в этих ячейках по-прежнему сохраняются, они просто недоступны для просмотра, а результаты вычисления остаются видимыми. Чтобы включить режим скрытия формул, выполните следующие действия.

1. Выделите ячейки, которые хотите скрыть.
2. В меню Формат выберите команду Ячейки.
3. На вкладке Защита окна диалога Формат ячеек установите флажок Скрыть формулы и нажмите ОК.
4. Выберите в меню Сервис команду Защита, а затем – Защитить лист.
5. В окне диалога Защитить лист установите флажок Содержимого и нажмите ОК.

Использование формул

Если бы не формулы, электронную таблицу можно было бы создать и с помощью текстового процессора. Формулы – это душа и сердце рабочего листа, и MS Excel предлагает богатые возможности для построения сложных формул. Напомним, что ввод формулы начинается со знака «=».

F3 fx =(C3/E3)*B3-D3

	A	B	C	D	E	F
1	Работники:	Кол-во прораб. дней	Оклад	Издержки	Кол-во раб.дней	Зарплата
2	Иванов А.В	25	7500	0	30	6250
3	Петров С.П.	14	5600	600	30	2013,333
4	Сидоров В.А.	21	60000	450	30	
5						

Рисунок 2 – Ввод данных с помощью формул

Использование ссылок в формулах

Ссылка является идентификатором ячейки или группы ячеек в книге. Создавая формулу, содержащую ссылки на ячейки, вы связываете формулу с ячейками книги. Значение формулы зависит от содержимого ячеек, на которые указывают ссылки, и оно изменяется при изменении содержимого этих ячеек.

Различают несколько типов ссылок. Относительная ссылка указывает на ячейку, основываясь на её положении относительно ячейки, в которой находится формула. Абсолютная ссылка использует для указания на ячейку её фиксированное положение на листе. Смешанная ссылка содержит относительную и абсолютную ссылку. Абсолютные и смешанные ссылки особенно полезны при копировании формулы из одного места листа в другое. Например, относительная ссылка на ячейку A1 записывается так: = A1; абсолютная ссылка на ячейку A1 имеет следующий вид: =$A$1; смешанные ссылки могут иметь вид: =$A1 и =A$1. Если символ доллара стоит перед

буквой, то координат столбца абсолютная, а строки – относительная. Если наоборот, тогда координат столбца – относительная, а строки – абсолютная.

С помощью клавиши F4 можно быстро изменить тип ссылки. Например:

1. Выделите ячейку A1 и введите:

=B1+B2

2. Нажмите клавишу F4, чтобы ссылку, находящуюся рядом с точкой вставки в строке формул, изменить на абсолютную. Формула примет вид:

=B1+$B$2

3. Снова нажмите F4, чтобы сделать эту ссылку смешанной (относительная координата столбца и абсолютная координата строки). Тогда формула примет вид:

=B1+B$2

4. Ещё раз нажмите F4, чтобы реверсировать эту смешанную ссылку (абсолютная координата столбца и относительная координата строки). Формула принимает вид:

=B1+$B2

5. Снова нажмите F4, чтобы вернуться к исходной относительной ссылке.

Вы можете ссылаться на другие листы той же книги так же легко, как и на ячейки текущего листа. Например, чтобы ввести в ячейку B10 листа Лист1 ссылку на ячейку A9 листа Лист2, выполните следующие действия:

1. Выделите ячейку B10 на листе Лист1 и введите знак равенства.
2. Щёлкните на ярлычке Лист2 в нижней части окна Книга1.
3. Щёлкните на ячейке A9 и нажмите клавишу Enter. После нажатия Enter снова будет активизирован Лист1 и в ячейке B10 появится формула:

=Лист2!A9.

Точно так же, как вы ссылаетесь на ячейки, расположенные на других листах текущей книги, вы можете ссылаться и на ячейки, находящиеся в другой книге. Такие ссылки называются внешними. При этом название книги, с которой вы собираетесь ссылаться, заключается в квадратные скобки, а листы и ячейки оформляются тем же самым образом. Например, ячейка A1 листа Лист1 книги Книга1 ссылается на ячейку A2 листа Лист2 книги Книга2:

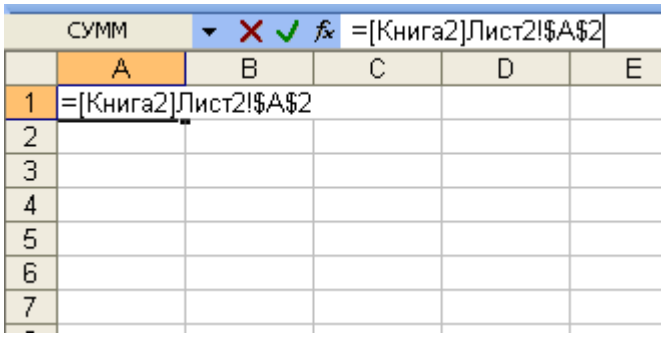


Рисунок 3 – Ссылки на листы других книг

Использование функций

Функция – это заранее определённая формула, которая оперирует с одним или несколькими значениями и возвращает значение (или значения). Многие функции Excel являются краткими вариантами часто используемых формул. Например, чтобы сложить ряд значений ячеек с помощью функции СУММ, достаточно просто выделить нужный диапазон.

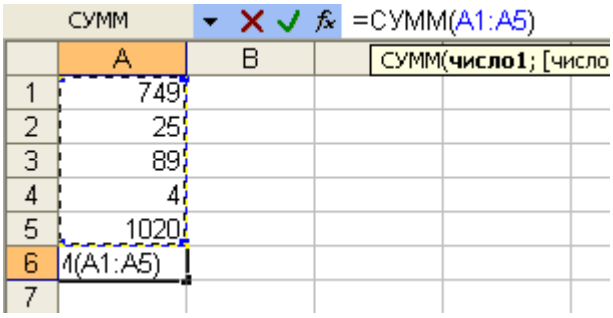


Рисунок 4 – Ввод данных, используя функцию СУММ

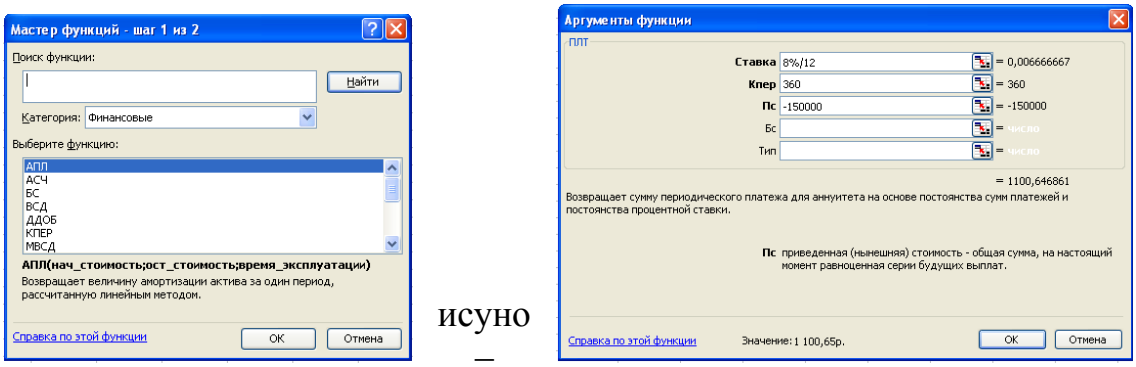
В Excel есть один из простых и удобных доступов к встроенным функциям – это вставка функции через кнопку Вставка функции. Например, чтобы вычислить размер выплат по ссуде с помощью функции ПЛТ, необходимо выполнить следующие действия:

1. Выделите ячейку.

2. Нажмите кнопку Вставка функции на стандартной панели инструментов.

3. В открывшемся окне диалога Мастер функций – шаг 1 из 2 в списке Категория выберите пункт Финансовые.

4. В списке Функция выберите ПЛТ и нажмите кнопку OK. После появления панели формул следуйте шагам, приведённым на рисунке 5.



исуно к 5

Окна диалога при работе с функцией ПЛТ

Работа с листами

Книга является подшивкой для всех своих листов, и подобно подшивке она может содержать листа разных типов, которые разрешаются вставлять, удалять или перемещать в любое место. В отличие от подшивки листы можно копировать, а также присваивать им имена или переименовывать их. Также можно выделять группу листов и редактировать их вместе – другими словами, позволяется вводить данные и применять форматы ко всем выделенным листам одновременно.

Разделение листов на области

Области окна являются одним из способов одновременного просмотра нескольких частей рабочего листа. Вы можете разделить любой лист вертикально, горизонтально или в обоих направлениях одновременно. Области окна предоставляют возможность синхронного прокручивания разных частей рабочего листа. Рассмотрим пример, для иллюстрации разбиения окна на области рассмотрим лист, представленный на рисунке 6.

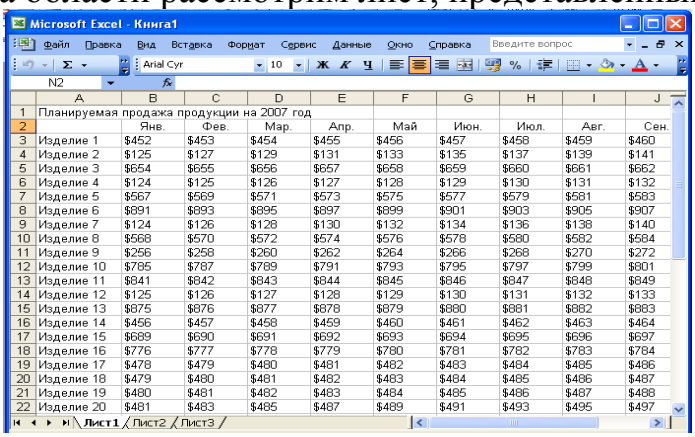


Рисунок 6 – Окно для разделения на области

В этом листе столбцы от B до M и строки от 3 до 43 содержат данные о продажах. В столбце N и строке 38 находятся итоги. Чтобы в листе видеть итоги в столбце N при работе с данными месячных продаж в столбцах от B до M, можно разделить окно на две области: одну в 7 столбцов и другую шириной в 1 столбец. Чтобы создать вертикальную область, установите указатель на маркере разделения по вертикали, представляющем собой узкий прямоугольник в правом конце горизонтальной полосы прокрутки. Когда указатель мыши находится на маркере разделения, он принимает форму перекрестия с двусторонней стрелкой как показано на рисунке 7.

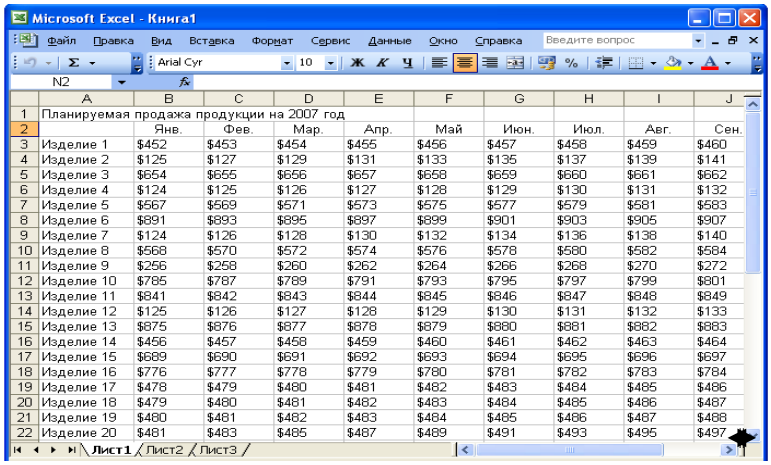
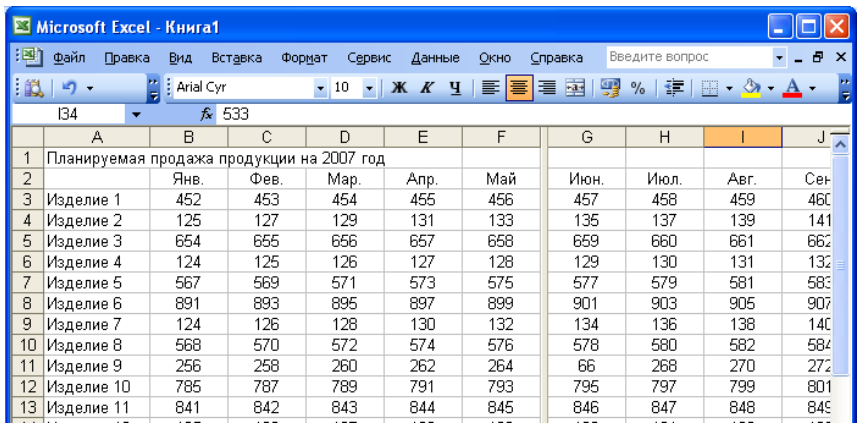


Рисунок 7 – Указатель мыши на маркере разделения

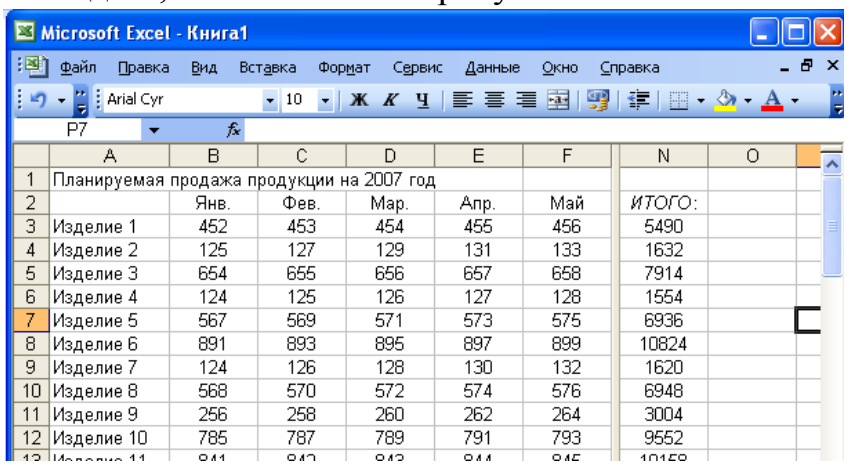
Если вы выделите любую ячейку в столбце G и дважды щёлкните на маркере разделения по вертикали, то лист будет выглядеть примерно так как показано на рисунке 8.



	A	B	C	D	E	F	G	H	I	J
1	Планируемая продажа продукции на 2007 год									
2		Янв.	Фев.	Мар.	Апр.	Май	Июн.	Июл.	Авг.	Сен
3	Изделие 1	452	453	454	455	456	457	458	459	460
4	Изделие 2	125	127	129	131	133	135	137	139	141
5	Изделие 3	654	655	656	657	658	659	660	661	662
6	Изделие 4	124	125	126	127	128	129	130	131	132
7	Изделие 5	567	569	571	573	575	577	579	581	583
8	Изделие 6	891	893	895	897	899	901	903	905	907
9	Изделие 7	124	126	128	130	132	134	136	138	140
10	Изделие 8	568	570	572	574	576	578	580	582	584
11	Изделие 9	256	258	260	262	264	66	268	270	272
12	Изделие 10	785	787	789	791	793	795	797	799	801
13	Изделие 11	841	842	843	844	845	846	847	848	849

Рисунок 8 – Окно, разделённое на две области

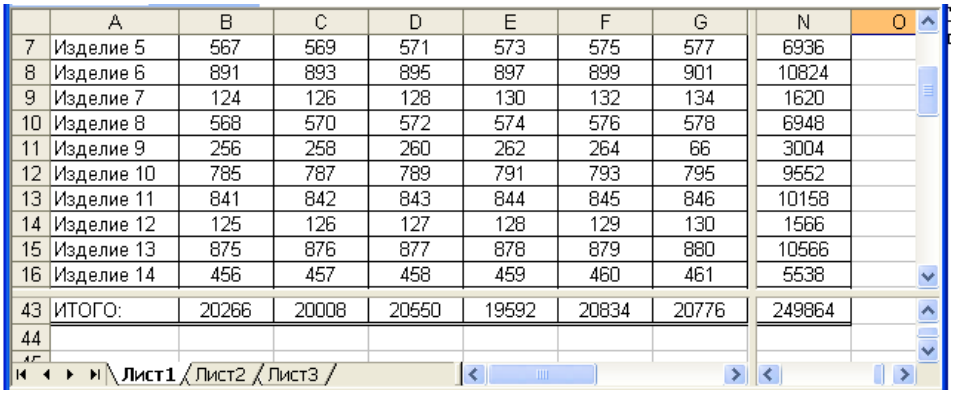
Теперь в окне будут две горизонтальные полосы прокрутки – по одной для каждой области окна. Затем используйте горизонтальную полосу прокрутки в правой области, чтобы прокрутить область до столбца N. После этого ваш лист должен выглядеть, как показано на рисунке 9.



	A	B	C	D	E	F	N	O
1	Планируемая продажа продукции на 2007 год							
2		Янв.	Фев.	Мар.	Апр.	Май	ИТОГО:	
3	Изделие 1	452	453	454	455	456	5490	
4	Изделие 2	125	127	129	131	133	1632	
5	Изделие 3	654	655	656	657	658	7914	
6	Изделие 4	124	125	126	127	128	1554	
7	Изделие 5	567	569	571	573	575	6936	
8	Изделие 6	891	893	895	897	899	10824	
9	Изделие 7	124	126	128	130	132	1620	
10	Изделие 8	568	570	572	574	576	6948	
11	Изделие 9	256	258	260	262	264	3004	
12	Изделие 10	785	787	789	791	793	9552	

Рисунок 9 – Вид окна после прокрутки правой области

Если вы хотите видеть месячные итоги в строке 43, можно создать горизонтальную область. Выделите любую ячейку в строке 43 и дважды щёлкните на маркере разделения по горизонтали. Ваш лист теперь должен выглядеть примерно так, как показано на рисунке 10.



	A	B	C	D	E	F	G	N	O
7	Изделие 5	567	569	571	573	575	577	6936	
8	Изделие 6	891	893	895	897	899	901	10824	
9	Изделие 7	124	126	128	130	132	134	1620	
10	Изделие 8	568	570	572	574	576	578	6948	
11	Изделие 9	256	258	260	262	264	66	3004	
12	Изделие 10	785	787	789	791	793	795	9552	
13	Изделие 11	841	842	843	844	845	846	10158	
14	Изделие 12	125	126	127	128	129	130	1566	
15	Изделие 13	875	876	877	878	879	880	10566	
16	Изделие 14	456	457	458	459	460	461	5538	
43	ИТОГО:	20266	20008	20550	19592	20834	20776	249864	
44									

Лист1 Лист2 Лист3

Рисунок 10 – Окно, разделённое на горизонтальные и вертикальные области

ТЕМА: **Основы программирование на языке Паскаль**

Программирование алгоритмов линейной структуры на языке Турбо Паскаль.

Программа на Паскале состоит из двух частей: описательной части и операторной части. Программа может начинаться с заголовка, состоящего из служебного слова Program и имени, имеющего синтаксис идентификатора. Операторная часть начинается служебным словом Begin, а заканчивается программа служебным словом End. После этого оператора ставится точка.
Синтаксис:

Program <Имя программы>;
<Описательная часть>
Begin
<Операторная часть>
End.

Операторы отделяются друг от друга точкой с запятой и располагаются по строкам произвольно

В интегрированной среде могут быть открыты сразу несколько окон для ввода программ. Переход между окнами осуществляется клавишей F6.

Комбинацией клавиш CTRL и F9 программа компилируется и запускается на выполнение.

Ввод и вывод информации осуществляется в окне ввода. Чтобы из окна программы перейти в окно ввода, нужно нажать комбинацию клавиш ALT F5. Перейти из окна ввода в окно программы можно, нажав любую клавишу.

Основным оператором процедурного языка программирования является оператор присваивания. Синтаксис оператора:

<Переменная>:=<Выражение>;

Переменная и выражение должны иметь один и тот же тип. Двоеточие и знак равенства записываются слитно и называются знаком присваивания.

Пример:

S := (a + b+ c)/3;
L := (S>0) OR (P<100);

В этом примере S и соответствующее выражение в правой части имеют вещественный тип, а L и выражение в правой части — логический тип.

Выражение записывается в строку, поэтому программист должен следить за последовательностью вычислений. Приоритет арифметических действий такой же, как в алгебре. Для изменения последовательности действий необходимо использовать скобки.

Пусть необходимо записать оператор присваивания для выражения

$$f = \frac{x^2 + 2y - \cos^2 x}{y^2 + \sqrt{2x^2 + 1}}$$

Оператор присваивания:

F := (x*x + 2 * y – SQR(COS(x)))/(y * y + SQRT(2 * x * x + 1))

Еще один пример:

$$k = \frac{e^{x+y}}{2x+3y}\sin^2 x^3$$

Оператор присваивания:

k := EXP(x+y)/(2*x+3*y)*SQR(SIN(x*x*x))

Со стандартными функциями Турбо Паскаля можно познакомиться, вызвав справочную систему интегрированной среды.

Вводить данные в Паскале можно при помощи двух стандартных процедур — Read и ReadLn. Процедуры используются следующим образом: сначала указывается имя процедуры, а затем в круглых скобках указывается список переменных, значения которых необходимо ввести. Переменные отделяются друг от друга запятыми.

Пример:

```
Read(x, y, z);
ReadLn(a, b, c);
```

При выполнении этих процедур вводимые значения отделяются друг от друга одним или несколькими пробелами или переходом на следующую строку. В отличие от Read выполнение процедуры ReadLn завершается только после нажатия на клавишу «возврат каретки».

Рассмотрим эту разницу на примере. При выполнении следующего фрагмента программы

```
Read(a, b);
Read(c);
```

и при вводе данных

```
4 5 1
```

переменной a присвоится значение 4, переменной b — 5, а переменной c — 1. В случае же фрагмента

```
ReadLn(a, b);
ReadLn(c);
```

при вводе данных таким же образом переменной a присвоится значение 4, переменной b — 5. После этого ожидается ввод значения c на новой строке. Значение 1 будет проигнорировано.

Часто процедура Readln без параметров ставится в конце программы для задержки окна ввода/вывода на экране до нажатия «возврат каретки». Однако если предыдущий ввод запрашивался процедурой Read, задержки не будет.

Вывод данных обеспечивается двумя стандартными процедурами Write и WriteLn. В скобках после имени процедуры могут быть записаны константы, выражения, переменные.

Пример:

```
Write('Корень уравнения= ', X);
Writeln('Сумма =',Sum,'  Среднее значение=',Sum/n);
```

Числа вещественных типов по умолчанию выводятся в экспоненциальной форме, что воспринимается обычным пользователем не лучшим образом. Для того, чтобы вывести число в общепринятой форме, задается формат вывода:

WriteLn(X:n:m);

где X — переменная вещественного типа, а n и m — выражения целого типа, характеризующие ширину поля вывода. Выражение n означает, что все число будет выравниваться по правому краю поля из n символов (недостающие символы слева заменяются пробелами), выражение m означает, что число будет выводиться с m знаками после запятой. При этом будет происходить округление абсолютной величины числа.

Для вывода целых чисел можно использовать форму:

WriteLn(a:n);

В этом случае выводимое значение будет выравниваться по правому краю поля шириной в n символов.

Отличие WriteLn от Write состоит в том, что после вывода всех значений, перечисленных в скобках, производится переход на следующую строку.

Алгоритм, в котором все действия выполняются последовательно одно за другим, называется линейным. Приведем пример программы, в которой реализуем линейный алгоритм.

Пример.

Условие задачи:

Какого роста цен за год можно ожидать, если правительство гарантирует, что инфляция в новом году составит p%?

Если за каждый месяц цены возрастут в 1 + p/100 раз, то за год рост цен составит $(1+p/100)^{12}$ раз или прирост в процентах

$$S = \left[\left(1 + \frac{p}{100}\right)^{12} - 1\right].100\%$$

Поясним формулу. За один месяц цены возрастут на p/100 и составят 1 + p/100. Для того, чтобы сократить изложение, обозначим эту величину за X.

$X = 1 + p/100$

Единица — это условная цена единица товара. В следующем месяце цены возрастут на тот же процент, но уже от имеющихся цен на остаток месяца: X* p/100. Цены будут иметь значение X + X * p/100 или X*(1 + p/100).

$X*(1 + p.100) = X^2$

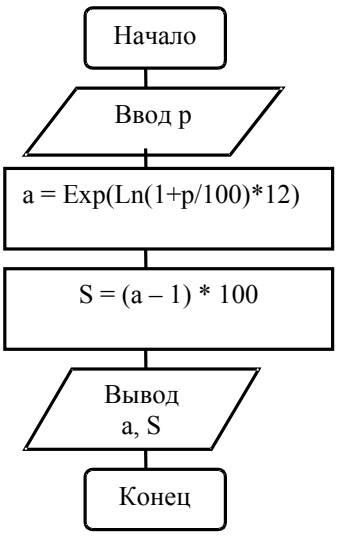
В следующем (третьем месяце) цены будут иметь значение:

$X^2 + X^2*p/100 = X^2*(1+p/100) = X^3$

И так далее. В конце 12-го месяца года цена единицы товара будет равна X^{12} или$(1 + p/100)^{12}$. Очевидно, что рост цены будет равен $(1 + p/100)^{12}$ – 1.

Чтобы получить прирост в процентах, необходимо умножить на 100%.

Схема алгоритма:



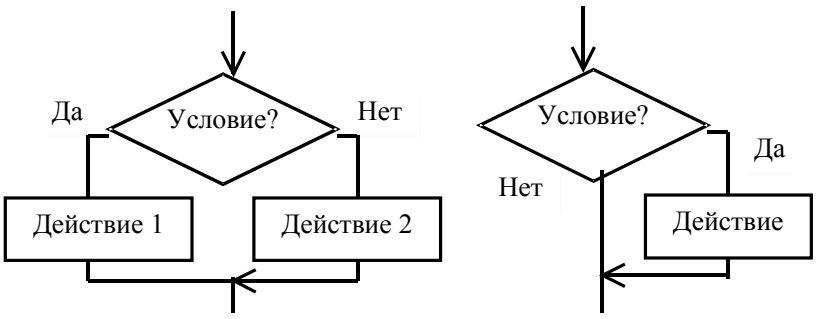
Программа на Турбо Паскале в этом случае может иметь следующий вид:

```
Program Procent;                          {Заголовок программы}
Var a, p, S: Real;                        {Описание переменных}
{p — процент инфляции в месяц.
a — кратность роста цен. S — рост цен за год в процентах}
Begin                                     {Начало операторной части }
Write('Введите процент месячной инфляции'); ReadLn(p);
                                          {Вывод подсказки ввода. Ввод переменной
p}
a := Exp(Ln(1+p/100)*12);                 {Оператор присваивания}
{Так как в Паскале нет возведения в степень, используется
тождество:
a^b = exp(Ln(a)*b)}
S := (a – 1) * 100;                       {Оператор присваивания}
Writeln('Годовой прирост цен составит  ', a:10:2,
 ' раз, или 'S:10:2, ' процентов');       {Процедура вывода}
End.                                      {Конец программы}
```

Программирование алгоритмов разветвляющейся структуры на языке Турбо Паскаль

Часть I. Изучение правил построения алгоритма разветвляющейся структуры и записи программы с использованием оператора If…Then…Else.

Алгоритм решения задачи, как правило, представляет собой совокупность стандартных алгоритмических структур. Одной из таких структур является развилка (полная и неполная). Разветвление применяется, когда в зависимости от условия нужно выполнить либо одно, либо другое действие.



Развилка полная Развилка неполная

Для программирования проверки условия и выбора действия в зависимости от условия используются условные операторы.

Условный оператор:

If <логическое выражение> Then <Действие 1> Else <Действие 2>;

Если логическое выражение имеет значение True, то выполняется <Действие 1>. Если логическое выражение имеет значение False, то выполняется <Действие 2>.

Под обозначением <Действие 1> и <Действие 2> понимается один любой оператор языка. Если в зависимости от условия потребуется выполнить несколько операторов, то такие операторы нужно заключить в операторные скобки Begin и End. В Турбо Паскале любая последовательность операторов, находящаяся между словами Begin и End, считается одним оператором, называемым составным оператором.

Условный оператор может использоваться без части Else. В этом случае реализуется структура «развилка неполная».

If <логическое выражение> Then <Действие>;

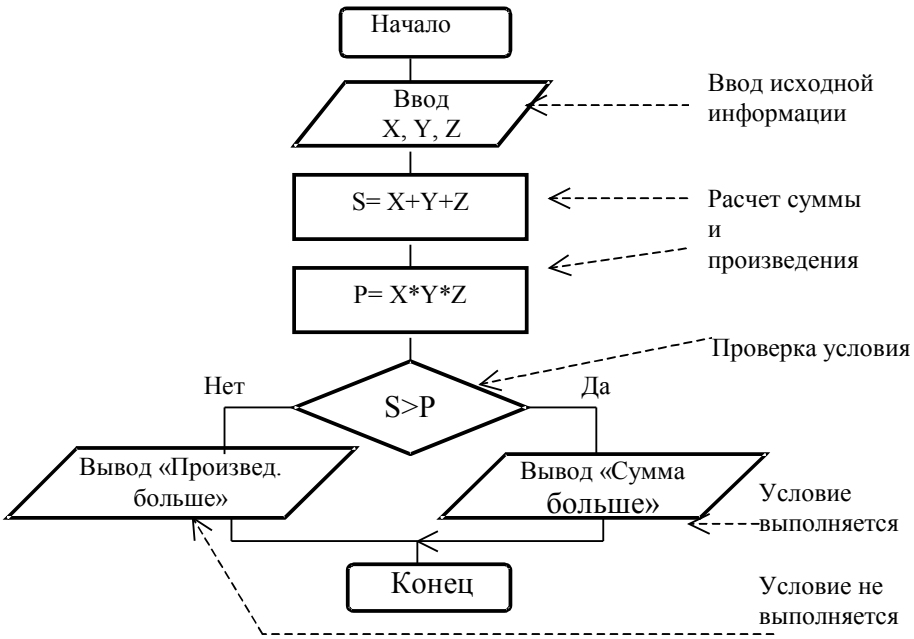
Если логическое выражение имеет значение True, то выполняется оператор, стоящий за служебным словом Then, иначе осуществляется переход к оператору, следующему за условным оператором.

Пример 1.

Условие задачи:

Даны числа X, Y, Z. Определить, что больше: сумма этих чисел или их произведение.

Схема алгоритма:



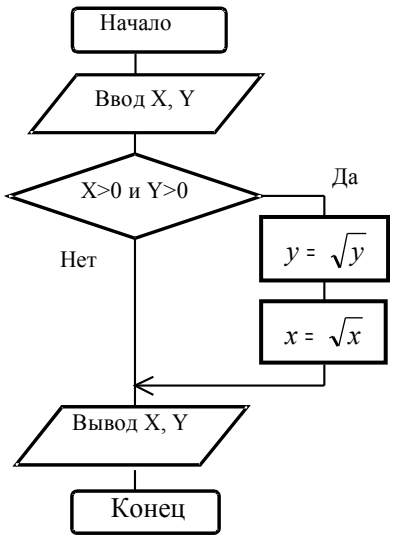
Программа:

```
Program Comparat;
Var x, y, z, S, P:Real;

Begin
Write('Введите X '); Readln(x);
Write('Введите Y '); Readln(y);
Write('Введите Z '); Readln(z);        {Ввод исходной информации}
S := x + y + z;
P := x * y * z;
If S > P Then WriteLn('Сумма больше, чем произведение X, Y, Z')
{Оператор, исполняемый в случае,
 если логическое выражение имеет значение True}
Else WriteLn('Произведение больше или равно сумме X, Y, Z');
{Оператор, исполняемый в случае,
 если логическое выражение имеет значение False}
End.
```

Пример 2.

Условие задачи:

Даны два числа X и Y. Вычислить квадратные корни данных чисел, если оба значения больше нуля, и оставить числа без изменения, если это не так.

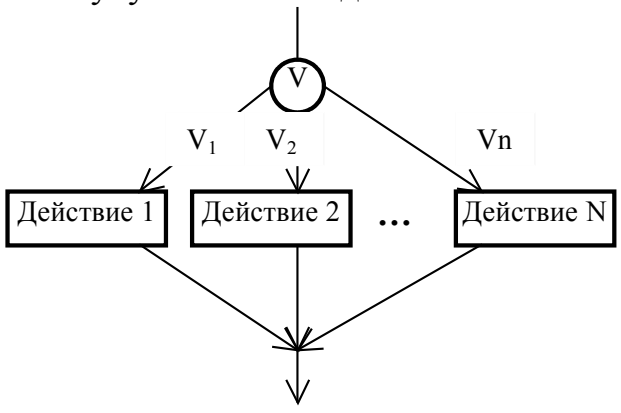


Программа:

```
Program XY;
Var x, y: Real;

Begin
Write('Введите x и y '); Readln(x, y);
If (x>0) and (y>0) Then Begin   {Если лог. выражение имеет значение True, то выполняется составной оператор}
X := Sqrt(x);
Y := Sqrt(y);
End;          {Условный оператор закончился}
Writeln('x=', x, ', y=', y);
Readln;
End.
```

Часть II. Изучение правил использования оператора Case…Of .

Если в алгоритме разветвляющейся структуры предполагается более двух вариантов (ветвей) расчета, а выбор варианта зависит от значения какой–либо одной переменной, то целесообразно использовать структуру «множественный выбор». Эта структура объединяет в себе несколько структур типа «развилка» и улучшает наглядность схемы алгоритма.



Решение задачи будет осуществляться по одной из ветвей алгоритма в зависимости от того, какое значение примет переменная V.

В программах такая структура реализуется с помощью оператора Case … Of.

Синтаксис оператора:

Case <Выражение порядкового типа> Of <Список выбора> Else <Оператор> End;

Где <Список выбора> — это одна или более конструкций вида:

<Константа или перечень констант> : <Оператор>;

Константы должны иметь такой же тип, что и выражение, следующее за служебным словом Case. Константы могут представлять собой интервал или разделяться запятыми.

Пример записи оператора Case:

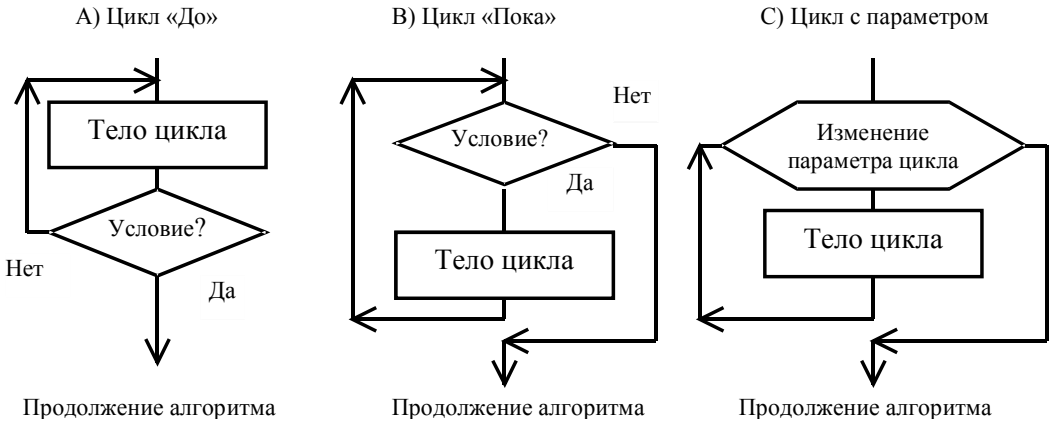
```
Case Ch Of
'+' : Z := X + Y;
'-' : Z := X - Y;
'*' : Z := X * Y;
'/' : Z := X / Y;

Else
Stop := True;
End;
```

Переменная Ch имеет символьный тип. «Список выбора» организуют символьные константы '+', '-', '*', '/'.

Часть Else может быть опущена.

Изучение организации циклических вычислений на примерах.

Цикл — типичная структура, характерная для программ, реализуемых на ЭВМ. Возможны три способа организации циклических структур алгоритмов:



Тело цикла — это повторяющаяся последовательность действий. Логический блок предназначен для управления циклом. Логический блок определяет количество проходов в цикле.

В цикле «До» условие окончания цикла расположено после тела цикла. Это означает, что тело цикла выполниться хотя бы один раз. В цикле «Пока» условие окончания цикла расположено до тела цикла, поэтому возможны варианты, когда цикл не выполнится ни разу.

В Паскале эти структуры реализуются с помощью оператора с постусловием и оператора с предусловием.

Синтаксис оператора с постусловием:

Repeat <последовательность операторов> Until <логическое выражение>;

Тело цикла расположено между служебными словами Repeat и Until. Это любая последовательность операторов языка. Операторы выполняются в цикле до тех пор, пока логическое выражение имеет значение False. Как только выражение примет значение True, цикл закончит свою работу и осуществится переход к выполнению следующего оператора, расположенного после оператора цикла.

Синтаксис оператора с предусловием:

While <логическое выражение> Do <оператор>;

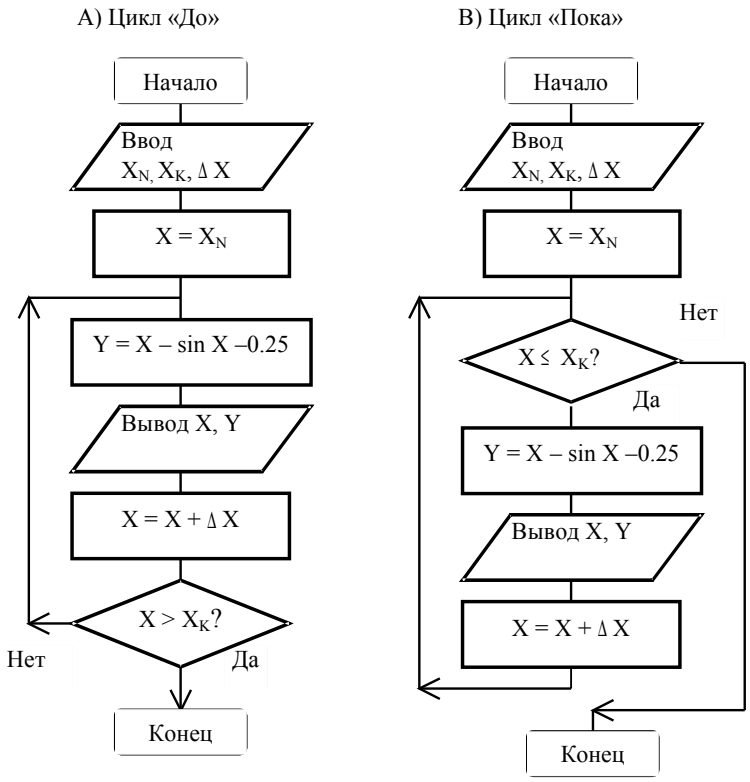
До тех пор, пока логическое выражение имеет значение True, выполняется тело цикла. Телом цикла может быть любой оператор языка, в том числе и составной.

Пример 1.

Условие задачи:

Написать программу табулирования функции Y = x – sin x – 0.25.

Схема алгоритма:



Переменная X изменяется в цикле и по значению этой переменной определяется условие окончания цикла.

Программа, организующая цикл с помощью оператора цикла с постусловием:

```
Program Tabl1;
Var x, xn, xk, dx, y: Real;
{x — текущее значение аргумента}
{xn — начальное значение аргумента}
{xk — конечное значение аргумента}
{dx — шаг изменения аргумента}
{y — значение функции}
Begin
Write('Начальное значение аргумента '); Readln(xn);
Write('Конечное значение аргумента '); Readln(xk);
Write('Шаг изменения аргумента '); Readln(dx);
x := xn;                    {Присвоили начальное значение аргументу}
Repeat
{Начало тела цикла}
Y := x – sin (x) – 0.25;    {Рассчитали значение функции}
Writeln ( x:8:2, y:10:2);   {Вывели результат на экран}
X := x + dx;                {Текущее значение аргумента увеличили на шаг}
{Конец тела цикла}
Until x>xk;                 {Проверка условия окончания цикла}
End.
```

Программа, организующая цикл с помощью оператора цикла с предусловием:

```
Program Tabl2;
Var x, xn, xk, dx, y: Real;
Begin
Write('Начальное значение аргумента '); Readln(xn);
Write('Конечное значение аргумента '); Readln(xk);
Write('Шаг изменения аргумента '); Readln(dx);
x:= xn;
While x<= xk Do  {Цикл выполняется до тех пор, пока выполняется условие}
 Begin           {В цикле выполняется один составной оператор}
Y := x – sin (x) – 0.25;
Writeln ( x:8:2, y:10:2);
X := x + dx;
 End;            {Конец тела цикла}
End.
```

Цикл с параметром в программах на Турбо Паскале реализуется с помощью оператора For ... To ... Do. Синтаксис оператора цикла с параметром:

For <идентификатор> := <выражение1> To <выражение2> Do <оператор>;

Идентификатор переменной и выражение должны иметь один и тот же порядковый тип. Переменная изменяется в цикле от значения <выражение1> до значения <выражение2> с шагом 1. Эта переменная управляет циклом и называется параметром цикла. В цикле выполняется один любой оператор языка, в том числе и составной. Шаг изменения параметра цикла постоянен и равен 1. Возможна другая интерпретация оператора:

For <идентификатор> := <выражение1> Downto <выражение2> Do <оператор>;

В этом случае шаг изменения параметра равен – 1.

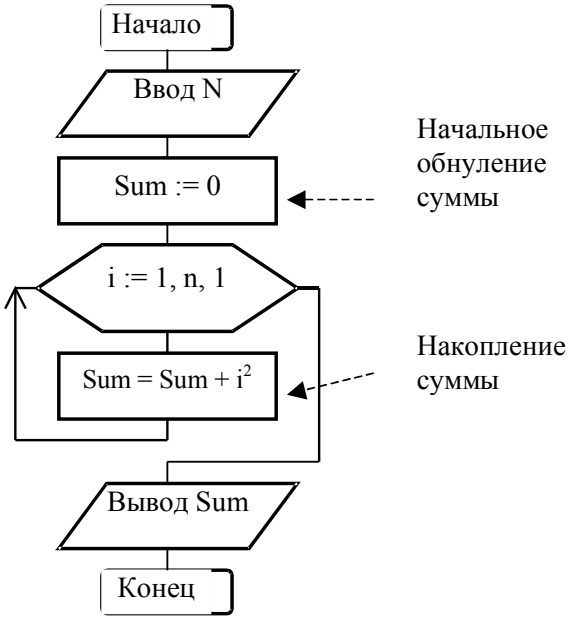
Оператор For чаще используется в случаях, когда при организации циклов необходимо использовать счетчик.

Пример 2.

Условие задачи:

Вычислить сумму квадратов первых N чисел натурального ряда.

Схема алгоритма:



Параметром цикла является переменная i. Значение переменной i изменяется от 1 до N с шагом 1. Каждое значение i, возведенное в квадрат прибавляется к переменной Sum и результат присваивается значению Sum. Таким образом, в переменной Sum накапливается сумма квадратов i. Так как при первом проходе значение Sum в правой части выражения должно быть равно нулю, до начала цикла значение Sum обнуляется. Тот же прием используется при подсчете произведения. Очевидно, что начальное значение произведения должно равняться 1.

Программа:

```
Program Summa;
Var i, n, Sum: integer;
Begin
Write('Введите n'); Readln(n);
```

```
Sum := 0;                    {Начальное обнуление суммы}
For i := 1 To n Do
Sum := Sum + i*i;
{Тело цикла — оператор присваивания для накопления суммы}
Writeln('Сумма =',Sum);
End.
```

Программирование алгоритмов циклической структуры на языке Турбо Паскаль. Итерационные алгоритмы

I. Изучение организации итерационных алгоритмов на примерах

Циклические вычисления могут иметь либо заранее известное количество шагов, либо они выполняются до достижения некоторого условия. Циклические вычисления с неизвестным числом повторений, в которых число повторений определяется требуемой точностью вычислений, называются итерационными вычислениями. Повторение последовательности операторов с проверкой условия в начале каждого прохода цикла называется итерацией.

Пример 1.

Условие задачи:

Рассчитать экспоненту с точностью ε путем разложения ее в ряд: $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$

Будем вычислять частичную сумму ряда правой части $S_n = \sum_{i=0}^{n} \frac{x^i}{i!}$ до тех пор, пока очередное слагаемое $\frac{x^i}{i!}$ не станет меньше погрешности ε.

Для вычисления каждого слагаемого ряда требуется возведение в степень и вычисление факториала (это дополнительный цикл). Часто в задачах для вычисления очередного слагаемого удобно рекуррентно использовать предыдущее слагаемое, а не организовывать дополнительный (внутренний) цикл. В данной задаче каждое очередное слагаемое можно рекуррентно вычислить через предыдущее: $U_i = \frac{X}{i} U_{i-1}$, что требует всего двух операций. Такая форма записи, в которой каждый следующий элемент расчета может быть получен из предыдущего, называется рекуррентной формой.

При построении таких алгоритмов полезно подстраховаться от зацикливания («вечного цикла»). Зацикливания могут возникнуть из–за ошибок в программе или вследствие накопления погрешностей. Чтобы избежать зацикливания, достаточно поставить лимит числа повторений цикла.

При построении алгоритма примем следующие обозначения:

S — искомая сумма, U — текущее слагаемое получаемое рекуррентно, n — определяет число шагов и используется для расчета факториала в

знаменателе слагаемого.

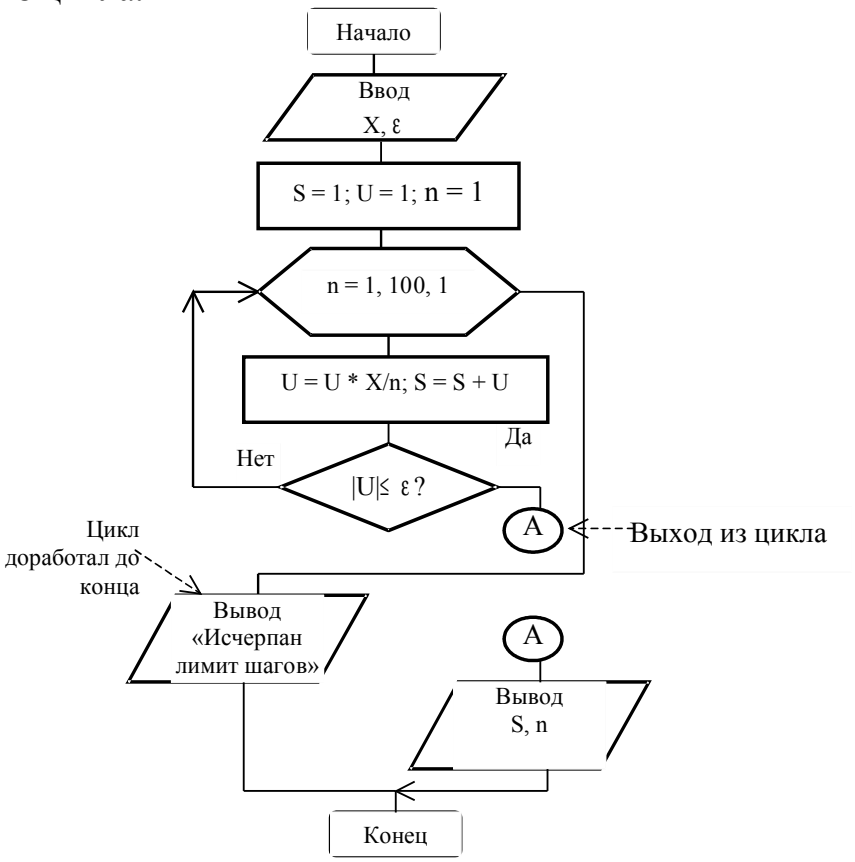
Схема алгоритма:

Программа:

```
Program Iteration;
Var x, Exp, S, U: Real; n: Integer;
Const Limit = 100;          {Предельное значение шагов задано константой}
Begin
Write('Задай аргумент и погрешность'); Readln(x, Eps);
S := 1;                     {Частичная сумма}
U := 1;                     {Первое слагаемое}
n := 1;                     {Число шагов}
Repeat
U := x / n * U;
S := S + U;
N := n + 1;
Until (abs(U) <= Eps) Or (n >= Limit);
If n>= Limit Then
Writeln(n, ' шагов не хватило для достижения заданной точности')
```

```
    Else
Writeln('Расчетное значение=', S:15:6, 'n=',n);
End.
```

Если для организации цикла в этой задаче использовать оператор цикла For, то в этом случае необходимо организовать принудительный выход из цикла. Предельное значение параметра цикла будет определяться лимитом числа повторений тела цикла. Как только выполнится условие $|U| \le \varepsilon$, следует выход из цикла.



Принудительный выход из цикла в программе на Паскале можно организовать, используя оператор перехода Goto. Синтаксис оператора:

Goto <Метка>;

После выполнения такого оператора, управление программой передается оператору, помеченному меткой. Меткой может быть идентификатор или целое число без знака до 9999.

Программа решения задачи:

```
Program Iteration;
Var x, Exp, S, U: Real; n: Integer;
Const Limit = 100;
Label M1, M2;                    {Метки должны быть описаны}
Begin
Write('Задай аргумент и погрешность'); Readln(x, Eps);
S := 1;                  U := 1;
For n := 1 To Limit Do
Begin
U := x / n * U; S := S + U;
```

```
If (abs(U) <= Eps) Then Goto M1;     {Выход из цикла}
End;
{Цикл доработал до конца}
Writeln(n, ‘ шагов не хватило для достижения заданной точности’);
Goto M2;
M1: Writeln(‘Расчетное значение =’, S:15:6, ‘n=’,n);
M2:End.
```

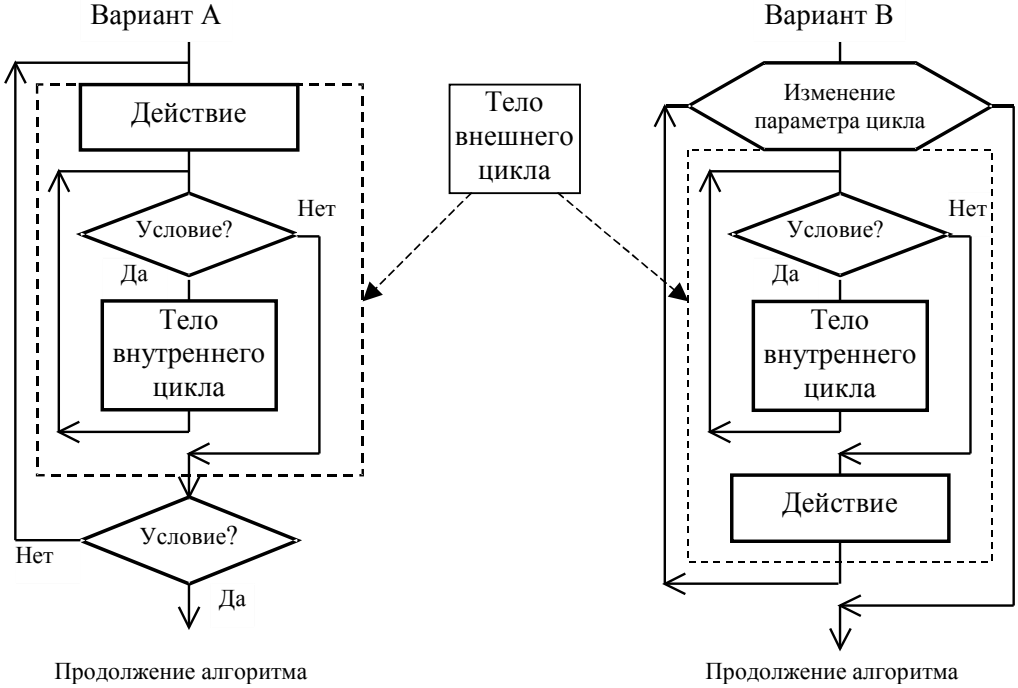
В программе используются два оператора перехода. Первый оператор перехода Goto M1 организует принудительный выход из цикла. Следующим выполняемым оператором будет оператор, помеченный меткой M1. После идентификатора метки ставится двоеточие. Второй оператор перехода Goto M2 организует обход вывода, ненужного в данном случае.

Программирование вложенных циклов

I. Изучение процесса алгоритмизации и программирования вложенных циклов на примерах.

Цикл внутри себя может содержать любые алгоритмические структуры. Если циклическая вычислительная структура содержит в себе другую циклическую структуру, то такие циклы называются вложенными. При использовании вложенных циклов необходимо составлять алгоритм таким образом, чтобы внутренний цикл полностью укладывался в циклическую часть внешнего цикла. При этом тип цикла («До», «Пока», «Цикл с параметром») не имеет значения. Внутренний цикл может в свою очередь содержать другой внутренний цикл.

Вот примеры структур вложенных циклов:



В варианте «А» цикл «До» содержит внутри себя цикл структуры «Пока». В варианте «В» внешним циклом является цикл с параметром, а внутренний цикл имеет структуру «Пока». Возможны и все другие сочетания

циклических структур.

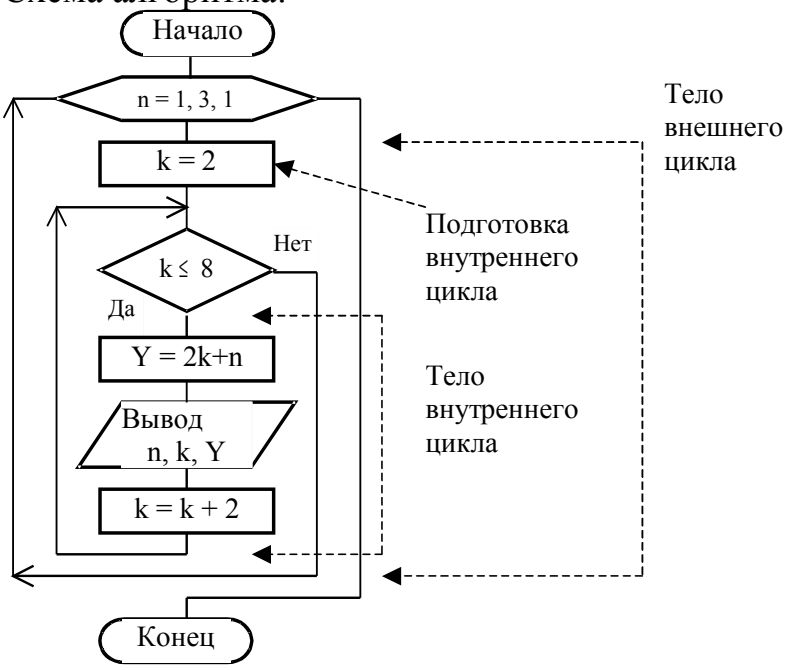
Структуру вложенных циклов рассмотрим на примере.

Пример 1.

Условие задачи:

Вычислить значение переменной Y = 2k + n при всех значениях переменных n = 1, 2, 3 и k = 2, 4, 6, 8.

Схема алгоритма:



Внешний цикл организован с помощью структуры «Цикл с параметром». Параметр n изменяется от 1 до 3. Для каждого n выполняется тело цикла, которое представляет собой один вычислительный блок (подготовка внутреннего цикла) и цикл структуры «Пока». Для каждого значения n полностью выполняется цикл по k (k побегает все значения от k = 2 до k = 8 с шагом 2).

Программа на Турбо Паскале:

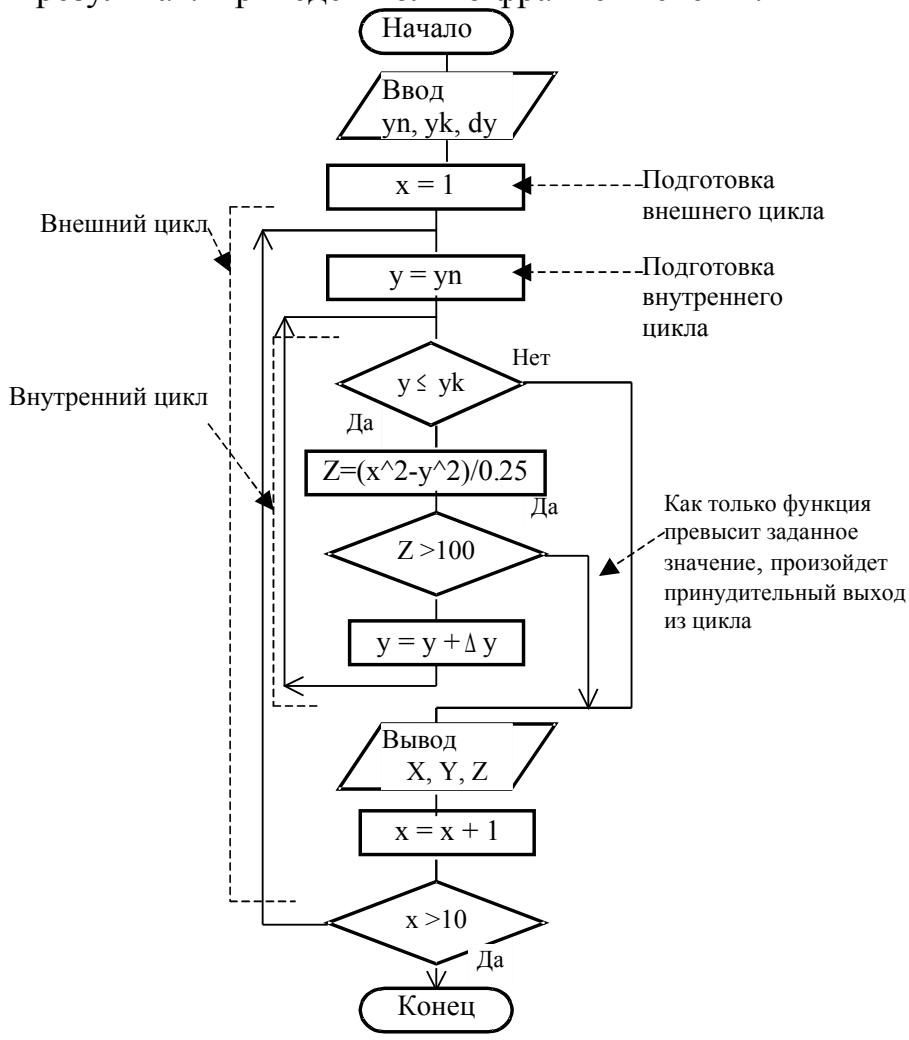
```
Program VlogCikl;
 Var n, k, Y: Integer;
Begin
For n := 1 To 3 Do              {Начало внешнего цикла}
Begin
K := 2;
While k <= 8 Do            {Начало внутреннего цикла}
Begin
Y := 2 * k + n;
Writeln(n:4, k:4, Y:4);
K := k + 2;
End;                            {Конец внутреннего цикла}
End;                            {Конец внешнего цикла}
End.
```

Пример 2.

Условие задачи:

Задана функция: $z = \frac{x^2 - y^2}{0.25}$. Для каждого X в интервале [1 … 10] с шагом 1 найти первое значение Y, при котором функция Z превысит значение 100.

Организуем два вложенных цикла: внешний цикл по X и внутренний по Y. При каждом значении X и Y рассчитаем значение функции. Когда Z примет значение большее 100, можно прервать продолжение цикла по Y и вывести результат. Приведем только фрагмент схемы.



Не трудно заметить, что вывод x, y, z будет производиться и в случае, когда цикл по Y доработает до конца. В задаче же требуется выводить только те значения, которые соответствуют условию Z >100, следовательно, необходимо еще раз проверить условие во внешнем цикле. В схеме алгоритма это не отображено. Этот недостаток исправлен в программе.

Организовать выход из цикла можно, воспользовавшись известным оператором безусловного перехода Goto. Для передачи управления чаще используются стандартные процедуры Exit, Halt, Break, Continue.

Процедура Exit осуществляет выход из блока (цикла или подпрограммы). Процедура Halt выполняет выход из программы. Процедура Break осуществляет выход из цикла. Continue осуществляет переход к следующей итерации, даже если предыдущая еще не завершена. При

программировании данной задачи, мы можем воспользоваться процедурой Break для того, чтобы выйти из внутреннего цикла до завершения его работы.

```
Program Cikl_Brk;
Var yn, yk, dy, x, y, z: Real;
Begin
Write('Y начальное?'); Readln(yn);
Write('Y конечное?'); Readln(yk);
Write('Шаг изменения Y?'); Readln(dy);
X := 1;
Repeat                                  {*Начало внешнего цикла}
y := yn;
While y <= yk Do                        {**Начало внутреннего цикла}
Begin
z := (x*x - y*y)/0.25;
y := y + dy;
If Z > 100 Then Break;                  {Принудительный выход из цикла}
{При выполнении условия следующим
исполняемым оператором будет — x := x +1}
End;                                    {**Конец внутреннего цикла}
X := x + 1;
If Z <= 100 Then Continue;              {Переход к следующей итерации}
{При выполнении условия оператор
 вывода не выполнится}
Writeln('X= ', (X-1):10:2,'    Y= ',(Y-dY):10:2,'   Z= ',z:10:2);
Until x>4;                              {*Конец внешнего цикла}
End.
```

В этой программе используется процедура Break для прерывания выполнения цикла и перехода к оператору, следующему за циклом. Процедура Continue используется для обхода оператора вывода, так как предназначен для перехода к следующей итерации цикла.

Алгоритмизация и программирование задач с использованием одномерных массивов

I. Изучение приемов программирования задач с использованием одномерных массивов

Массив представляет собой набор однотипных элементов. Переменная массив имеет одно имя (например, a), а каждый элемент массива имеет свой порядковый номер, называемый индексом (a[1], a[2], a[3], …, a[n]).

Массив, в котором положение элемента определяется одним индексом, называется одномерным. В математике аналогом такой структуры является вектор $\vec{a}$.

Индексом может быть произвольное выражение порядкового типа, заключенное в квадратные скобки. Например, Sum[True], Vect[i + 1].

Обработка массивов производится путем изменения индексов компонент. То есть имеется способ перехода от одного элемента массива к другому путем изменения индекса. Тот факт, что индекс может быть вычисляемым объектом, выделяет массив среди многих других структур данных.

Как и любая переменная в Турбо Паскале, массив должен быть описан.

Массивом называют как тип данных, так и переменную этого типа. Пример описания массивового типа:

Type SomeArr = Array [1 .. 100] Of Real;

Var a, b, c : SomeArr;

Мы описали тип данных массив. Имя типа — SomeArr. Индексы в этом типе — переменные, выражения или константы целого типа. Индексы могут изменяться в интервале от 1 до 100. Каждый элемент массива имеет вещественный тип. В разделе описания переменных описаны переменные массивового типа a, b, c.

Массив можно описать сразу в разделе описания переменных:

Var a, b, c: Array [1 .. 100] Of Real;

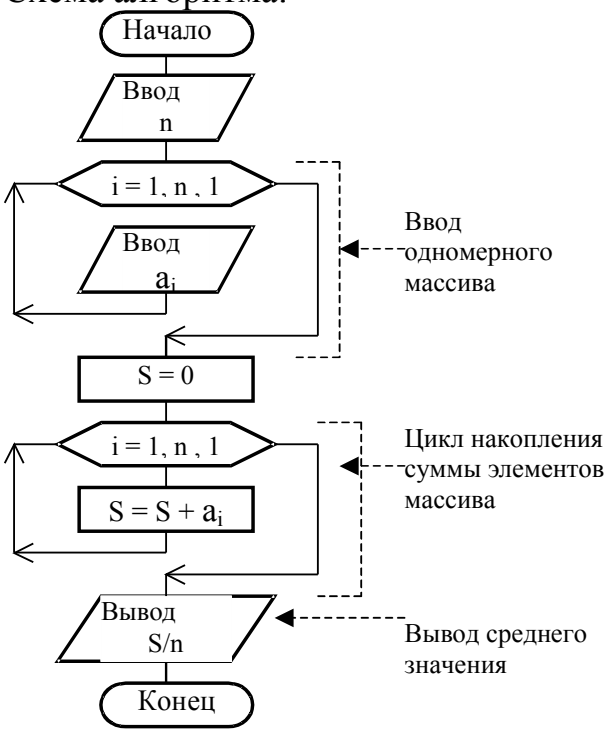
Рассмотрим практические примеры работы с массивами.

Пример 1.

Условие задачи:

Написать программу, которая вычисляет среднее арифметическое значений элементов одномерного массива.

Схема алгоритма:



При обработке массивов часто используется структура «цикл с параметром», так как индекс массива представляет собой счетчик и изменяется с шагом, равным 1.

Program mid;

{*Программа вычисляет среднее арифметическое значений*

```
элементов массива}
Const nn = 10;
Var
        a: Array[1..nn] Of Integer;
        I, n: Integer;
        S: Real;
Begin
{Ввод количества элементов массива}
        Write('Количество элементов?'); Readln(n);
{Ввод элементов массива}
        For i := 1 To n Do
        Begin
                Write('Введите a[' , i, ']'); Readln(a[i]);
        End;
{Вычисление суммы элементов массива}
S := 0;
        For i := 1 To n Do S := S + a[i];
{Вывод результата на экран}
        Writeln('Результат: ', (S/n):5:3);
End.
```

В разделе констант определена константа nn, значение которой используется для определения граничного индекса. Использование раздела констант позволяет сгруппировать в начале программы величины, характерные для конкретного примера.

Массив a описан в разделе описания переменных. Это массив целых чисел. Индекс элементов массивы изменяется в интервале от 1 до nn. Если в программе на Турбо Паскале требуется обрабатывать массивы переменной размерности, то приходится описывать массивы с максимальным возможным в данной задаче числом элементов, а затем использовать только часть из этих элементов в каждом конкретном случае.

Переменная i используется в качестве параметра оператора цикла. Переменная n — количество элементов массива, исходная величина. S — результат вычислений. В этой переменной накапливается сумма элементов массива.

Рассмотрим отдельно фрагмент ввода массива.

```
        Begin
                Write('Введите a[' , i, ']'); Readln(a[i]);
        End;
```

Ввод массива (и вывод тоже) необходимо производить поэлементно. Ввести массив — это значит ввести все его элементы. Индекс элемента массива должен пробежать все значения от 1 до n. Для этого организуем цикл, в котором индекс массива является параметром цикла. Оператор For здесь особенно уместен. Оператору ввода предшествует оператор вывода, с помощью которого осуществляется вывод подсказки

Для расчета суммы используется прием накопления суммы.

```
S := 0;
    For i := 1 To n Do S := S + a[i];
```

Этот прием заключается в том, что в формуле вычисления суммы в правой и левой частях выражения записано имя одной и той же переменной. Начальное значение должно быть равно 0. В цикле к текущему значению суммы S прибавляется значение очередного элемента массива, и результат записывается в переменную S, которая для следующего прохода цикла становится текущим значением.

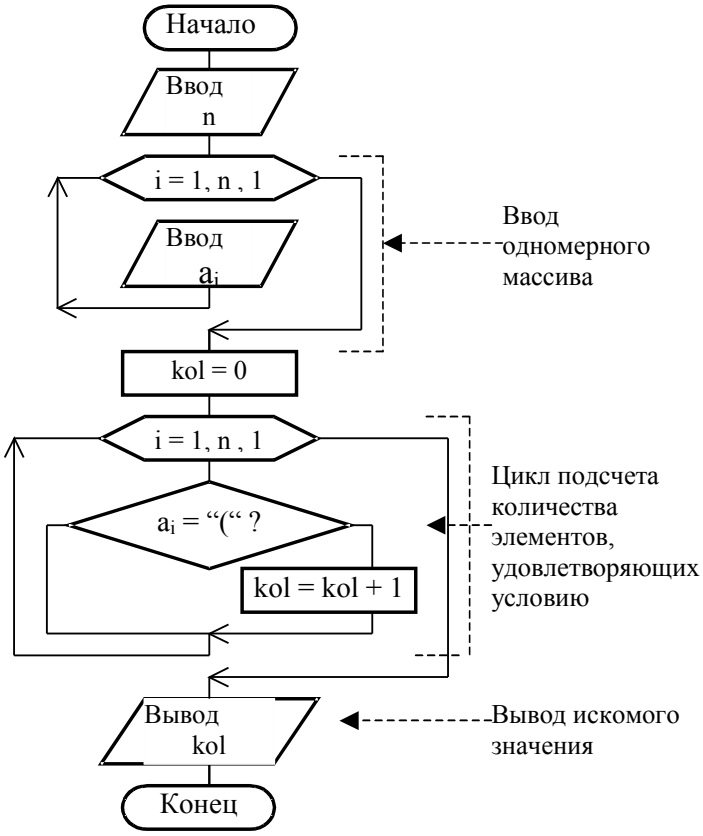
По окончании работы оператора цикла в переменной S накопится сумма всех элементов массива. Для определения среднего значения полученную сумму следует разделить на количество элементов массива и вывести результат на экран.

Пример 2.

Условие задачи:

В массиве символов подсчитать количество символов, представляющих собой открывающую скобку «(».

Схема алгоритма:



Программа:

```
Program Count_1;
Const nn = 100;
Var a: Array[1..nn] Of Char; i, n, Kol: Integer;
Begin
    Write('Количество элементов?'); Readln(n);
    For i := 1 To n Do
    Begin
```

```
            Write('Введите a[' , i, ']'); Readln(a[i]);
        End;
Kol := 0;
        For i := 1 To n Do
If a[i] = '(' Then Kol := Kol + 1;
        Writeln('Результат: ', Kol:3);
End.
```

Переменная a представляет собой массив символов, максимальное количество которых может быть 100. Индекс массива задан граничной парой или диапазоном. Индекс массива может изменяться от 1 до 100. Компоненты массива — символы. Переменная i служит параметром цикла. n — входная переменная, определяющая количество элементов массива в каждом конкретном случае. Kol — результат вычислений.

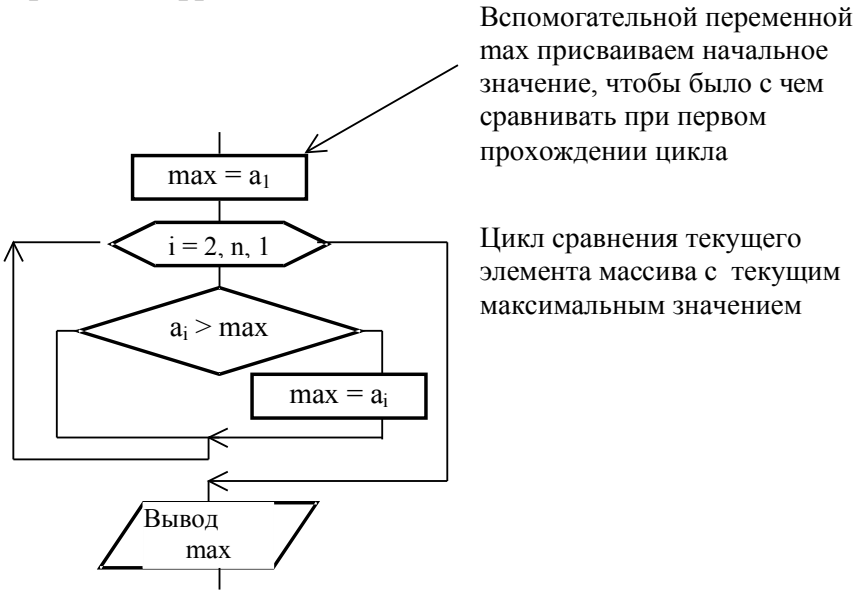
В переменной Kol накапливается значение количества элементов. Эти элементы массива a удовлетворяют условию a_i ='('. Суммируются единицы, которые в результате дадут нам значение общего количества элементов, отвечающих заданному условию.

Пример 3.

Условие задачи:

Найти максимальный элемент последовательности, состоящей из n элементов.

Приведем фрагмент схемы:



Программа:

```
Program max;                          {Поиск максимального элемента}
Const nn = 100;
Type list = 1..nn;
Var i : list; max, n : Integer;
a : Array [list] Of Integer;
Begin
```

Write('Количество элементов?'); Readln(n); {*Ввод количества элементов массива*}

```
For i := 1 To n Do                    {Ввод элементов массива}
Begin
Write('Введите ',i,'—й элемент массива '); Readln(a [i])
end;
max := a[1];
For i:= 2 To n Do
If a[i] > max Then max := a[i];
Write ('Максимальный элемент= ', max)
end.
```

Тип данных list, заданный в разделе описания типов, используется в качестве граничной пары в описании массива. Вспомогательная переменная i, которая используется в качестве параметра цикла, имеет такой же тип, как и индекс массива. Это обеспечивает дополнительный контроль соответствия параметра цикла индексу массива.

Переменная max, принимает значение первого элемента массива a. Над всеми остальными элементами массива выполняется однотипная операция: элемент массива сравнивается с переменной max. Если встречается элемент массива больший max, то max приобретает значение этого элемента. Если условие a_i > max не выполняется, то присваивания не происходит. По окончании цикла в переменной max останется значение самого большого элемента.

Пример 4.

Условие задачи:

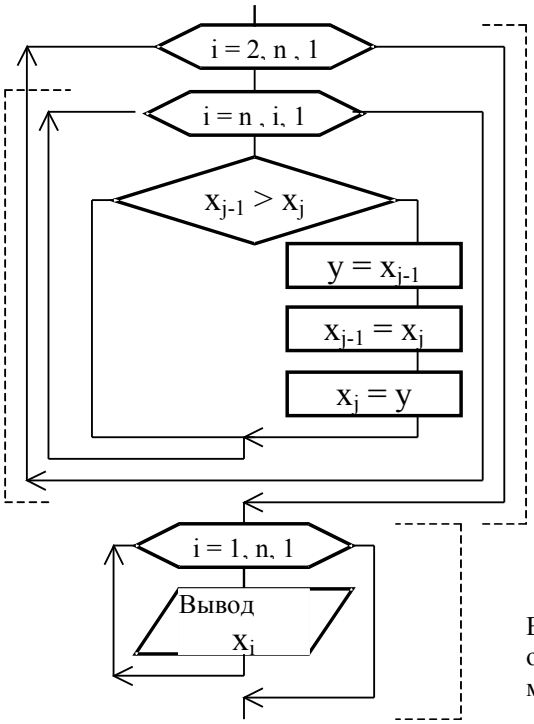
Разработать алгоритм и написать программу сортировки одномерного массива по возрастанию.

Для решения задачи воспользуемся алгоритмом, который получил название «метод пузырька». Этот алгоритм заключается в следующем.

Перебираются все элементы массива и производятся сравнения: первый элемент со вторым второй с третьим, и так далее до n–го. Если значение некоторого элемента больше значения следующего элемента, то эти элементы меняются местами.

В результате последовательность может быть не отсортирована. Однако после первого прохода можно с уверенностью сказать, что наибольшее значение находится на своем месте — на правой границе массива. Выполнив еще один проход по n – 1 элементам, получим два упорядоченных значения. Если осуществить n – 1 проход, получим упорядоченный массив.

Фрагмент схемы алгоритма:




```
Program puzirok;
{Упорядочивание элементов массива методом «пузырька»}
Const nn = 100;
Type mass = Array [1 .. nn] Of Real;
Var i, j, n : Integer; y : Real; x : mass;
Begin
{Ввод исходной информации}
Read(n);
For i := 1 To n Do
Begin
Write('Введите ',i,'–й элемент массива '); Readln(x[i])
End;
{Упорядочивание массива по возрастанию}
For i := 2 To n Do                                  {Проход}
Begin
For j := n Downto i Do                              {Цикл для сравнения}
If x[j – 1] > x[j] Then
Begin
y := x[j – 1]; x[j – 1] := x[j]; x[j] := y  {Перестановка}
End;
End;
{Вывод результата — упорядоченного массива}
For i:= 1 To n Do Write(x[i]:8:2)
End.
```

Этот алгоритм можно реализуется с помощью двух циклов For — внутреннего и внешнего. Число повторений внешнего цикла на 1 меньше числа элементов массива (так как последовательность из одного значения

проверять нет смысла). Число повторений внутреннего цикла последовательно сокращаться от n до 1.

Внешний цикл по i определяет количество проходов. Цикл по j создан для сравнения пар элементов массива. Сравниваются два элемента, расположенных рядом. Если $X_{J-1} > X_J$, то осуществляется перестановка элементов местами. Для того чтобы не потерять значение одного из переставляемых элементов, вводится вспомогательная переменная Y, которая представляет собой буфер для хранения x[j-1].

Для вывода массива, так же как и для ввода, требуется организовать цикл, определяющий индекс каждого элемента массива. Вывести массив — значит вывести все его элементы. Выводится X_i, при этом i пробегает все значения от 1 до n с шагом 1.

Работа с двухмерными массивами

Изучение приемов программирования задач с использованием двухмерных массивов

В Турбо Паскале имеется возможность работы с двухмерными, трехмерными массивами, а также массивами большей размерности. Наиболее часто используются двухмерные массивы.

В математике аналогом двухмерного массива является матрица.

$$A = \begin{bmatrix} 1 & 1 & 0 & 3 \\ 3 & 3 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

Матрица A состоит из четырех строк и четырех столбцов. В матрице каждый элемент идентифицируется номером строки и номером столбца, на пересечении которых он расположен.

В двухмерном массиве элемент также определяется двумя индексами, которые заключаются в квадратные скобки и отделяются друг от друга запятыми, например

a[2, 3]

В двухмерном массиве первый индекс указывает на номер строки, а второй индекс указывает на номер столбца, на пересечении которых расположен элемент. В матрице A, записанной выше, элемент a[2, 3] равен 1, а элемент a[3, 2] равен 0.

Предположим, перед нами стоит задача обнулить все элементы первой строки. Можно рассматривать строку матрицы как одномерный массив. Тогда нужно записать оператор:

```
For i:=1 To 4 Do a [1, i ] :=0;
```

Первый индекс элемента остается неизменным, он равен 1. Это номер строки. Второй индекс изменяется от 1 до 4. Это номер столбца.

Пусть теперь необходимо обнулить все элементы первого столбца.

```
For i:=1 To 4 Do a [i ,1 ] :=0;
```

В цикле организуем изменение первого индекса. Второй индекс, определяющий номер столбца остается неизменным.

Если нужно обнулить все элементы матрицы, то следует организовать два вложенных цикла.

```
For i:=1 To 4 Do
For j:=1 To 4 Do
a [i ,j] :=0;
```

При каждом значении i (номера строки) j пробегает все значения от 1 до 4.

Так, при работе с двумерными массивами для определения каждого элемента массива нужно организовать два вложенных цикла. Например, для ввода значений элементов матрицы, состоящей из 4-х строк и 4-х столбцов, нужно записать следующий код:

```
For i:=1 To 4 Do
  For j:=1 To 4 Do
Readln(a [i, j]);
```

Перед оператором ввода можно сделать «подсказку»:

```
For i:=1 To 4 Do
  For j:=1 To 4 Do
Begin
Write('Введите a[' , i, ',', j,']'); Readln(a [i, j]);
End;
```

Ввод по строкам, реализованный в приведенном фрагменте программы является наиболее естественным. В случае необходимости можно организовать заполнение массива по столбцам. Для этого внешний цикл должен быть организован по номеру столбца (j), а внутренний — по номеру строки (i) с соответствующими границами изменения индексов.

```
For j:=1 To m Do
  For i:=1 To n Do
Begin
Write('Введите a[' , i, ',', j,']'); Readln(a [i, j]);
End;
```

Двухмерный массив описывается так же, как и одномерный массив, но границы индексов должны быть указаны для двух его размерностей, например

```
Var A: array [1..4, 1..4] Of Real;
```

Номер строки массива A может принимать значения в интервале от 1 до 4. Номер столбца массива A также может принимать значения в интервале от 1 до 4.

Пользователь может описать тип «массив» в разделе описания типов, а затем объявить переменную этого типа:

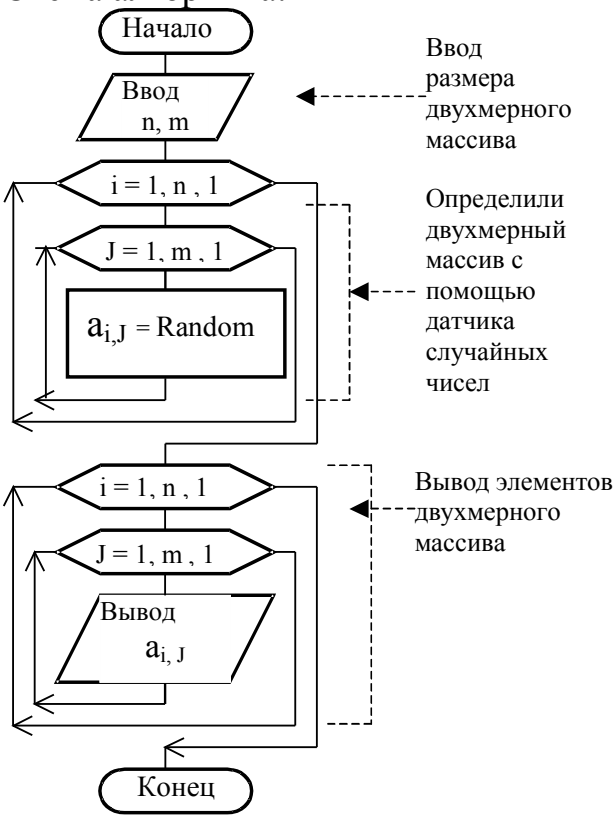
```
Type Mass = array [1..4, 1..4] Of Real;
Matrix = array [1..5, 1..6] Of integer;
Var A: Mass; B: Matrix;
```

Типы Mass и Matrix созданы для описания двухмерных массивов. Описание типа индекса представляет собой список диапазонов изменения индексов двухмерного массива. Элементы массива, заданного с помощью типа Matrix, будут иметь целый тип. Элементы массива, заданного с помощью типа Mass, будут иметь вещественный тип.

Пример 1.

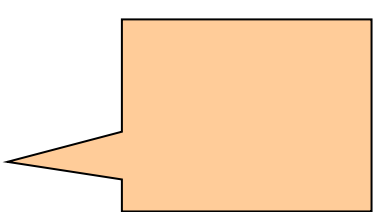
Условие задачи: Сформировать матрицу размером N на M и вывести ее на экран.

Схема алгоритма:



Программа:

```
Program Mass_2;
{Программа формирует двухмерный массив с помощью датчика случайных чисел и выводит массив на экран}
Const nn = 10; mm = 10;
Var
        a: Array[1..nn, 1..mm] Of Real;
        I, J, n, m: Integer;
Begin
{Ввод размера массива}
        Write('Количество строк матрицы?'); Readln(n);
        Write('Количество столбцов матрицы?'); Readln(m);
{Задали элементы массива}
For i:=1 To n Do
  For j:=1 To m Do
Begin
a [i, j] := Random;
```



```
End;
{Вывод результата на экран}
For i:=1 To n Do
Begin
For j:=1 To m Do
Write(a [i, j]:8:3);
Writeln;
End;
End.
```

Вывести массив — значит вывести все элементы массива. Также как и при вводе массива, следует организовать два цикла по строкам и по столбцам для определения индексов элементов массива. Элементы массива A выводятся оператором Write. Это означает, что курсор после очередного вывода остается в той же строке. Элементы массива выводятся в той же строке до тех пор, пока работает цикл по j. Как только цикл по j завершится, пустой оператор Writeln переведет курсор на новую строку. Такая организация вывода позволяет представить информацию в виде матрицы.

Пример 2.

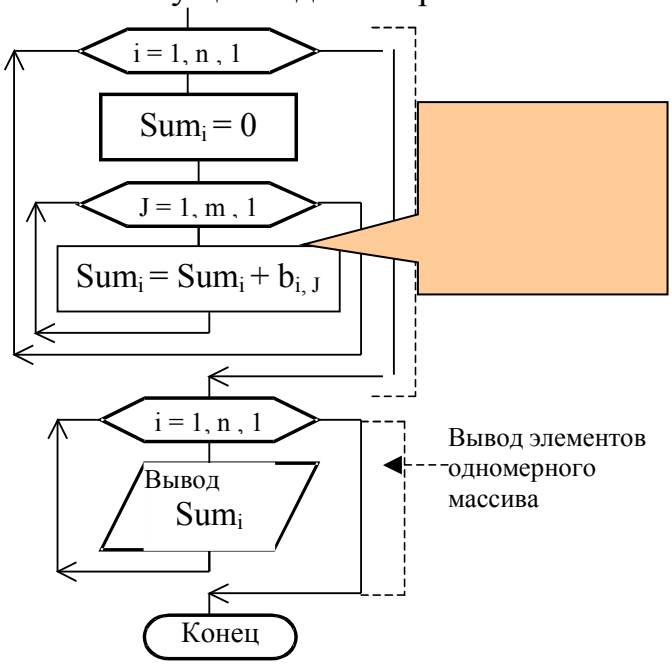
Условие задачи: Рассчитать суммы элементов строк матрицы. Результат представить в виде одномерного массива.

Дано: n — количество строк матрицы. m — количество столбцов матрицы.

[B] — матрица с элементами b_{ij}, где i изменяется от 1 до n, j изменяется от 1 до m.

Найти: $Sum_i = \sum_{j=1}^{m} b_{ij}$

Схема алгоритма для ввода матрицы аналогична схеме определения массива в предыдущей задаче, поэтому представим только фрагмент схемы для решения текущей задачи. Фрагмент схемы:



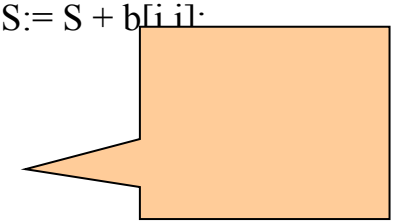
Программа:

```
Program SumStrok;
Type Mass = Array [1..10] Of Real;
Var I, J, n, m: Integer;
Sum: Mass; B: Array [1..10] Of Mass; {Описали массив массивов}
Begin
Write('Количество строк матрицы?'); Readln(n);
Write('Количество столбцов матрицы?'); Readln(m);
For i:=1 To n Do
For j:=1 To m Do
Begin
Write('Введите b[' , i, ',', j,']'); Readln(b[i, j]);
End;
For i:=1 To n Do
Begin
Sum[i] :=0;
For j:=1 To m Do Sum[i] := Sum[i] + b[i,j];
End;
For i:= 1 To n Do WriteLn ('Сумма элементов строки ', i, '= ',Sum[i] : 10:3);
End.
```

Сумма каждой строки матрицы должна храниться в соответствующем элементе одномерного массива. Так, сумма 1-ой строки должна храниться в первом элементе Sum_1, сумма 2-ой строки в элементе Sum_2 и так далее. Цикл по i определяет номер элемента одномерного массива Sum и номер строки матрицы B. Индекс j определяет номер строки матрицы. В цикле по j накапливается сумма элементов соответствующей строки матрицы.

На поиск в массиве элемента, с заданными значениями индексов, затрачивается время. (Адрес i-го элемента определяется прибавлением к адресу начала массива значения i). Поэтому для повышения эффективности лучше использовать вспомогательную переменную S при суммировании, что исключает многократное обращение к элементам массива Sum.

```
For i:=1 To n Do
Begin
S:=0;

For j:=1 To m Do S:= S + b[i,j];
Sum[i] := S;
End;
```



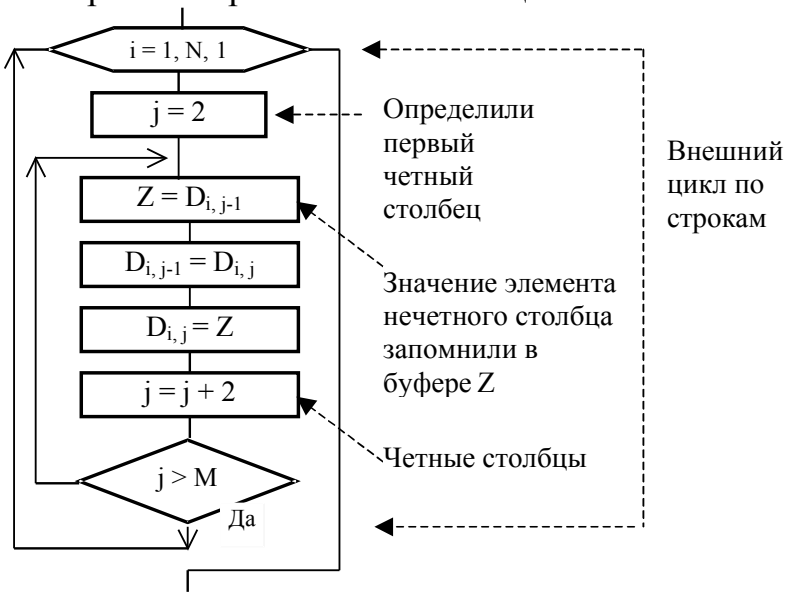
Пример 3.

Условие задачи:

Дана матрица размером N на M. Поменять местами четные и нечетные

столбцы матрицы.

Схемы ввода и вывода матрицы приведены в предыдущих примерах. Схема алгоритма перестановки столбцов в соответствие с условием задачи:



Для перестановки элементов четных и нечетных столбцов организован цикл с шагом 2. Внутри цикла по строкам организован цикл структуры «До». Перебираются только четные столбцы (со второго по последний с шагом 2). Нечетный столбец определен как J – 1. Во избежание потери информации вводится вспомогательная переменная Z, которая является буфером для хранения значения элемента нечетного столбца.

Программа:

```
Program Change_Stolb;
Var i, j, n, m: Integer; Z: Real;
D: Array [1..10, 1..10] Of Real;
Begin
{Ввод размера матрицы}
Write('Количество строк матрицы?'); Readln(n);
Write('Количество столбцов матрицы?'); Readln(m);
{Ввод матрицы}
For i:=1 To n Do
For j:=1 To m Do
Begin
Write('Введите D[' , i, ',', j,']'); Readln(D[i, j]);
End;
{ ***Перестановка столбцов ***}
For i:=1 To n Do          {Цикл по строкам}
Begin
j := 2;                   {Подготовка цикла по столбцам}
Repeat                    {Цикл по столбцам}
{Перестановка четного и нечетного элементов}
Z := D[i, j - 1]; D[i, j - 1]:= D[i, j]; D[i, j] := Z;
j := j + 2;               {Перебираем только четные столбцы}
```

```
Until j > m;
End;
{Вывод результирующей матрицы}
For i:=1 To n Do
Begin
For j:=1 To m Do
Write(D [i, j]:8:3);
Writeln;
End;
End.
```

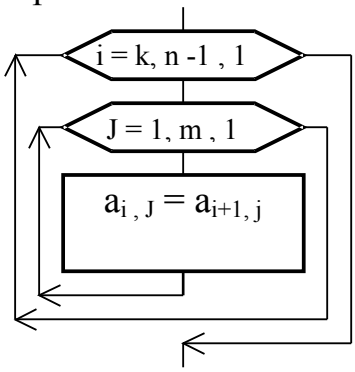
Пример 4.

Условие задачи:

Написать программу удаления строки из матрицы.

Удалить k-ую строку матрицы можно, сдвинув строки, следующие за k-ой строкой на одну позицию вверх.

Фрагмент схемы:



Фрагмент программы:

```
For i:= k To n – 1 Do
For j:= 1 To m Do
        a[i, j] := a[i + 1, j];
```

Все строки, начиная с (k+1)-й, нужно переместить вверх. Цикл по строкам начинается с k-ой строки. Все элементы k-ой строки заменяются на k + 1 строку. Число строк уменьшается на единицу.

Пример 5.

Условие задачи:

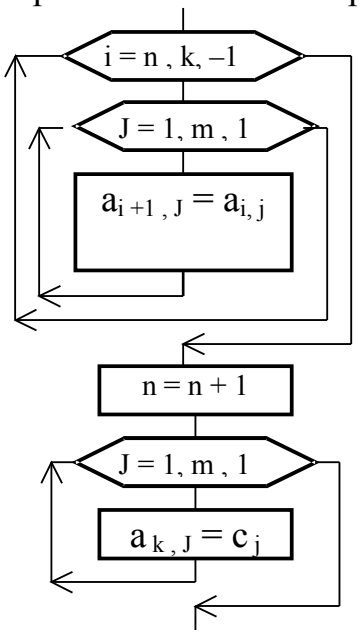
Написать программу включения строки в матрицу. Включаемая строка задана как вектор.

В этой задаче задана матрица размером N на M и вектор, состоящий из M элементов. Задано k — номер новой строки.

Фрагмент программы (описательная часть):

```
Const nn=20; mm=40;
Var a: array [1..nn+1, 1..mm] Of Real;
```

c: array [1..mm] Of Real;
I, j, n, m, k: Integer;
Фрагмент схемы алгоритма:



Передвинули элементы матрицы на одну строку вниз

В k – ую строку записали элементы массива С

Фрагмент программы (исполняемая часть):

```
For I := n DownTo k Do
For j:= 1 To m Do
a[i + 1, j] := a[i, j];          {с n –ой по k –ую строки матрицы передвинули вниз}
N := n+1;
For j:= 1 To m Do
a[k, j] := c[j];                 {На место k –ой строки матрицы записали вектор}
```

Матрица [A] будет увеличена на одну строку, поэтому в описании переменной [A] максимальный размер строк увеличен на единицу.

Два первых цикла организуют перемещение строк матрицы вниз на одну строку.

В цикле по строкам параметр цикла изменяется с шагом –1. Начиная с n-ой строки и до k-ой, все строки смещаются вниз, оставляя k-ую строку свободной для элементов вектора С. Перемещение одной строки связано с пересылкой всех элементов этой строки, что требует организации цикла по номеру столбца. Отдельно организован цикл для записи элементов вектора С на место k-ой строки матрицы [A]. В цикле по j элементам A_{kj} присваивается значение C_j

Работа со строковыми переменными

Изучение приемов программирования по обработке данных строкового типа

1.1. Строковый тип и действия, которые можно производить с переменными строкового типа.

Для обработки текстов в Турбо Паскале используется строковый тип.

Строковая константа широко использовалась в комментариях для списков вывода оператора Write. Строковая константа — это совокупность любых символов, заключенная в апострофы, например: ‘Количество отрицательных элементов вектора =’, ‘Введите n’, ‘Нажмите <Enter>’ и т.д.

Переменная строкового типа описывается с помощью служебного слова String, например

```
Var
S: String;
Str_simv: String[20];
```

В описании типа String длина строки может принимать любое целое значение от 1 до 255. Если длина не указана, то берется максимальная длина, равная 255. Переменная Str_simv может хранить строки длиной не более 20 символов. Описание String без указания длины отводит для переменной место, как и описание String[255].

Можно описать строковый тип, а затем этот тип использовать для описания переменных:

```
Const N=10;
Type
Str = String[20]; Str_1 = String[N];
Var
Value_a, Property_b: Str;
. . .
```

Для переменной типа String[N] выделяется N+1 байт памяти.

S[0] — хранит код, соответствующий числу символов в строке.

Ord(S[0]) — длина строки S.

Значением строки может быть любая последовательность символов, заключенная в одинарные кавычки:

‘abcde-абвгд’

‘123 ? 321’

Строки можно присваивать, объединять и сравнивать.

```
S := ‘Кон’ +’кат’ + ‘ена’ + ‘ция’;
```

Сложение строк производится с помощью оператора конкатенации. Этот оператор записывается с помощью знака +. При выполнении действия сложения начало строки, идущей после знака “+”, подсоединяется к концу строки, указанной до этого знака.

Произвольные пары строк могут сравниваться с помощью операторов отношений: = , <> , < , > , <=, >= .

‘Abcd’ < ’abcd’

Сравнение происходит посимвольно слева направо. При этом символ 1 считается меньше символа 2, если его код в таблице ASCII меньше, чем у символа 2. Код заглавной буквы «А» меньше, чем у прописной буквы «а».

Результат сравнения этих двух строк (Abcd с заглавной буквой «A» и abcd с прописной “a”) есть «правда».

Обработка текстов имеет некоторое сходство с обработкой массивов. Символы в тексте (как и элементы в массиве) упорядочены по номеру позиции, которую они занимают, и переход к следующему символу легко осуществляется изменением номера позиции на 1. Поэтому многие типовые алгоритмы обработки массивов в несколько модифицированном виде могут использоваться и при обработке текстов.

1.2. Стандартные процедуры и функции

При работе со строками используется набор стандартных процедур и функций.

Процедура/функция	Действие	Примеры
Length(S : String): Byte;	Возвращает текущую длину строки S (число символов, из которых состоит строка S).	Stroka := ‘Пример’; L := Length(Stroka); L равно 6.
Copy(S : String; Start, Len: Integer) : String;	Результат — строковая величина, состоящая из Len символов строки S, начиная с символа номер Start.	Str := Copy(‘Function’, 2, 4); Результат: ‘unct’. Str := Copy(‘Сарай’, 3, 3); Результат: ‘рай’.
Pos(Subs, S : String) : Byte;	Ищет вхождение подстроки Subs в строке S и возвращает номер первого символа Subs в S или 0 , если S не содержит Subs.	i := Pos(‘о’, ‘Космодром’); Результат: 2. p := Pos(‘одр’, ‘Космодром’); Результат: 5.
Insert(Var S : String; Subs : String; Start : Integer);	Вставляет подстроку Subs в строку S , начиная с позиции Start.	S := ‘Начало–конец’; Insert(S, ‘середина–’, 8); Результат: ‘Начало–середина–конец’.
Delete(Var S : String; Start, Len : Integer);	Видоизменяет строку S, удаляя Len символов, начиная с номера Start.	S := ‘Строка’; Delete(S, 2, 4); Результат: ‘Са’.
Str(X [: Width [: Dec]]; Var S : String);	Преобразует численную величину X в последовательность символов, соответствующей значению величины, затем преобразует эту	Str(45, S); Результат: ‘45’. Str(45:3, S); Результат: ‘␣45’.

	последовательность в строковую величину, которую присваивает переменной S.	
Процедура преобразования строки в число: Val(S: String; Var V; Var ErrCode : Integer);	Преобразует строковое выражение S в число и присваивает его значение переменной V. Если преобразование возможно, то переменная ErrCode равна нулю, в противном случае она содержит номер символа в S, на котором процедура застопорилась. Тип V должен соответствовать содержимому строки S. Если в S имеется точка, то V должна быть вещественного типа.	Пусть Var IntVar, Rep : Integer; Процедура: Val(‘23’, IntVar, Rep); IntVar = 23. Rep = 0 Процедура: Val(’23.0’, IntVar, Rep); IntVar = неизменно. Rep = 3.

3. Примеры решения задач по обработке строковых переменных

Пример 1.
Условие задачи:
Выделить и напечатать первую строку текста.
Программа:

```
Var V, T : String; P1 : integer;
Begin
Write(‘Введите текст’); Readln(T);
P1 := Pos(#13, T);
V := Copy(T, 1, P1 – 1);
Writeln(V);
End.
```

Функция Pos определит номер позиции заданного символа–разделителя в исходном тексте T.

С помощью функции Copy выделяется цепочка символов от начала текста до позиции разделителя строк P1. Выделенный фрагмент является первой строкой текста. Оператор Writeln(V) выведет первую строку на экран.

Пример 2.
Условие задачи:

Распечатать исходный текст по строкам. Строки в тексте разделены символом–разделителем.

Приведем один из вариантов решения задачи:

```
Const Q = #13;
Var V, G, T : String; P : Integer;
Begin
Write('Введите текст'); Readln(T); G := T;
P := Pos(Q, T);
While P <> 0 Do
Begin
V := Copy(T, 1, P – 1); Writeln(V);
Delete(T, 1, P);
P := Pos(Q, T);
End;
Writeln(T);
End.
```

С помощью оператора цикла While организуем повтор действий:

Копируем из исходного текста подстроку, ограниченную первым символом и символом–разделителем. Позиция символа–разделителя определена в переменной P. Распечатываем подстроку. Удаляем эту строку из исходного текста. Процедура Delete удаляет из строки T P символов, начиная с первого. С помощью функции Pos определяем позицию следующего символа–разделителя. Цикл продолжается до тех пор, пока P — позиция символа–разделителя не станет равной 0.

Возможно, после последнего символа–разделителя останется часть текста, если предположить, что весь текст необязательно заканчивается символом–разделителем. В этом случае эту часть текста необходимо вывести. Для этого предназначен последний оператор вывода Writeln(T).

Задача может быть решена значительно проще. Можно организовать цикл по всей длине текста и выводить текст посимвольно в строку, используя оператор вывода с переходом на новую строку только в том месте, где встречается разделитель строк.

```
For i := 1 To Length(T) Do
If T[I] = Q Then Writeln Else Write(T[I]);
```

Пример 3.

Условие задачи:

Разделить текст на строки, содержащие не более Col символов, делая перенос на другую строку на месте пробела.

Эта задача отличается от предыдущей тем, что символом разделителем является пробел, но не каждый пробел, а только тот что наиболее приближен к позиции Col каждой строки.

Если посимвольно исходный текст сравнивать с пробелом, причем последняя позиция сравнения равна величине Col (максимальное число символов в строке), то можно определить позицию последнего пробела.

```
For i := 1 To Col Do
If T[i] = ' ' Then P := i;
```

Программа:

```
Program Prim_3;
Const Col = 20;
Var V, G, T : String; P, i : integer;
Begin
Write('Введите текст'); Readln(T);
G := T; {Сохраним исходный текст в переменной G}
While Pos(' ', T) <>0 Do
{До тех пор, пока в тексте имеются пробелы
(позиция пробела в тексте не равна 0) выполняем}
Begin
For i := 1 To Col Do      {поиск ближайшего к позиции Col пробела }
If T[i] = ' ' Then P := i;
{копирование подстроки и вывод ее на экран}
V := Copy(T, 1, P); Writeln(V);
{формирование нового текста, в котором предыдущая строка вырезана}
T := Copy(T, P+1, Length(T));
End;
Writeln(T);
End.
```

Если за последним пробелом текст продолжается, то он будет выведен оператором вывода, расположенным после цикла.

Пример 4.

Условие задачи:

Распечатать исходный текст таким образом, чтобы каждое слово было напечатано на отдельной строке.

Слово — это последовательность букв, заключенная между символами, не являющимися буквами. Чаще всего слова отделяются друг от друга пробелами. Необходимо найти очередной символ–разделитель, тогда часть текста, заключенная между позициями двух соседних разделителей, будет словом.

```
k := 1;                                   {Начало первого слова}
For i := 1 To Length(T) Do
If T[i] = ' ' Then Begin
Writeln(Copy(T, k, i – k));       {Вывод слова на новой строке}
k := i + 1;                       {Начало нового слова}
End;
```

k — символ, с которого начинается слово. Первое слово начинается с 1–й позиции. Будем считать, что первый символ не пробел. В цикле каждый символ строки сравниваем с пробелом. Если очередной символ T[i] равен пробелу, то часть строки T (подстрока с k–ого по (i–k)–й символы) выводится

на экран с помощью оператора Writeln. K принимает новое значение. (i+1) — это позиция, следующая за пробелом, т.е. начало нового слова.

Пример 5.

Условие задачи:

Выяснить, является ли буква гласной или согласной.

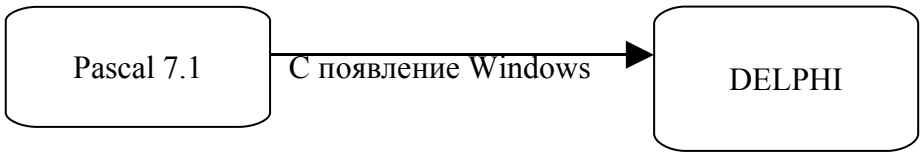
Чтобы выяснить является буква гласной или согласной, нужно сравнить ее с элементами подготовленного заранее массива гласных или согласных. Массив гласных предпочтительнее — требует меньше памяти.

```
Const Str_1 = ‘йуеыаоэяиюё’;
Var a : Char;
. . .
If Pos(a, Str_1) <>0 Then Writeln(‘Гласная’)
Else Writeln(‘Согласная’);
```

В разделе описания констант опишем строковую константу. Эта строковая константа содержит в качестве своих символов только гласные. В тексте программы достаточно проверить номер позиции буквы в строке Str_1. Если буква совпадает с каким–либо элементом массива гласных, т.е. позиция вхождения этой буквы в строку неравна 0, то это означает, что буква гласная.

ТЕМА: **Введение в Delphi.**

После освоения программирования на языке Паскаль можно переходить к программированию в среде Delphi. Дело в том, что в системе Delphi для написания кода используется язык Object Pascal, которой является дальнейшим развитием языка Turbo Pascal.



Delphi и Windows используют графический интерфейс. Delphi опирается на возможности Windows.

Очень важен процесс сохранения программы. Запишем процедуру по сохранению программы. 1) Создание вашей папки. 2) В ней создаётся папка с именем будущей программы. 3) В созданной папке сохраняем все файлы программы.

Система Delphi это:

1. Объектно-ориентированное программирование.
2. Система визуального программирования.
3. Система событийно-ориентированного программирования, т.е. реакция объекта на любое событие описывается в отдельной процедуре. Нет процедуры, нет реакции. Данные процедуры называются ОБРАБОТЧИКАМИ.

Структура интерфейса Delphi состоит из нескольких разделов.

Панель управления: на любой кнопке имеется пиктограмма, т.е. небольшой рисунок.

Палитра компонент: содержит различные вкладки. Любая из вкладок фактически является панелью инструментов, состоящая из кнопок. Любая кнопка представляет заготовку для экранных объектов создаваемого проекта.

Форма: представляет заготовку для окна будущей программы. Именно на форме размещаются объекты программы.

Список всех объектов: список представлен в форме древовидной структуры.

Инспектор объектов: в одной вкладке содержится свойства объекта, в другой события.

TEMA: Страница Standard

Страница Standard содержит ряд часто используемых компонентов общего назначения. На странице Standard палитры компонентов сосредоточены стандартные для Windows интерфейсные элементы.

Frame - рама. Наравне с формой служит контейнером для размещения других компонентов. В отличие от формы может размещаться в палитре компонентов, создавая заготовки компонентов. Впервые введен в версию Delphi 5.

MainMenu - главное меню программы. Компонент способен создавать и обслуживать сложные иерархические меню.

PopupMenu - вспомогательное или локальное меню. Обычно это меню появляется в отдельном окне после нажатия правой кнопки мыши.

Label - метка. Этот компонент используется для размещения в окне не очень длинных однострочных надписей.

Edit - строка ввода. Предназначена для ввода, отображения или редактирования одной текстовой строки. **Memo** - многострочный текстовый редактор. Используется для ввода и/или отображения многострочного текста.

Button - командная кнопка. Обработчик события OnClick этого компонента обычно используется для реализации некоторой команды.

CheckBox - независимый переключатель. Щелчок мышью на этом компоненте в работающей программе изменяет его логическое свойство Checked.

RadioButton - зависимый переключатель. Обычно объединяется как минимум еще с одним таким же компонентом в группу. Щелчок по переключателю приводит к автоматическому освобождению ранее выбранного переключателя в той же группе.

ListBox - список выбора. Содержит список предлагаемых вариантов (опций) и дает возможность проконтролировать текущий выбор.

ComboBox - комбинированный список выбора. Представляет собой комбинацию списка выбора и текстового редактора.

ScrollBar - полоса управления. Представляет собой вертикальную или горизонтальную полосу, напоминающую полосы прокрутки по бокам Windows-окна.

GroupBox - группа элементов. Этот компонент используется для группировки нескольких связанных по смыслу компонентов.

RadioGroup - группа зависимых переключателей. Содержит специальные свойства для обслуживания нескольких связанных зависимых переключателей.

Panel - панель. Этот компонент, как и GroupBox, служит для объединения нескольких компонентов. Содержит внутреннюю и внешнюю кромки, что позволяет создать эффекты “вдавленности” и “выпуклости”.

Actiontist - список действий. Служит для централизованной реакции программы на действия пользователя, связанные с выбором одного из группы однотипных управляющих элементов таких как опции меню, пиктографические кнопки и т. п. Впервые, введен в версии Delphi 4.

TEMA: **Страница Additional**

В страницу Additonal помещены 18 дополнительных компонентов, с помощью которых можно разнообразить вид диалоговых окон.

BitBtn - командная кнопка с надписью и пиктограммой.

SpeedButton - пиктографическая кнопка. Обычно используется для быстрого доступа к тем или иным опциям главного меню.

MaskEdit - специальный текстовый редактор. Способен фильтровать вводимый текст, например, для правильного ввода даты.

StringGrid - таблица строк. Этот компонент обладает мощными возможностями для представления текстовой информации в табличном виде.

DrawGrid - произвольная таблица. В отличие от StringGrid ячейки этого компонента могут содержать произвольную информацию, в том числе и рисунки.

Image - рисунок. Этот компонент предназначен для отображения рисунков, в том числе пиктограмм и метафайлов.

Shape - фигура. С помощью этого компонента вы можете вставить в окно правильную геометрическую фигуру - прямоугольник, эллипс, окружность и т. п.

Bevel - кромка. Служит для выделения отдельных частей окна трехмерными рамками или полосами.

ScrollBox - панель с полосами прокрутки. В отличие от компонента Panel автоматически вставляет полосы прокрутки, если размещенные в нем компоненты отсекаются его границами.

CheckListBox - список множественного выбора. Отличается от стандартного компонента ListBox наличием рядом с каждой опцией независимого переключателя типа CheckBox, облегчающего выбор сразу нескольких опций. Впервые введен в версии 3.

Splitter - граница. Этот компонент размещается на форме между двумя другими видимыми компонентами и дает возможность пользователю во время прогона программы перемещать границу, отделяющую компоненты друг от друга. Впервые введен в версии 3.

StaticText - статический текст. Отличается от стандартного компонента Label наличием собственного windows-окна, что позволяет обводить текст рамкой или выделять его в виде “вдавленной” части формы. Впервые введен в версии 3.

ControiBar - полоса управления. Служит контейнером для “причаливаемых” компонентов в технологии Drag&Dock. Впервые введен в версии 4.

ApplicationEvents - получатель события. Если этот компонент помещен на форму, он будет получать все предназначенные для программы сообщения Windows (без этого компонента сообщения принимает глобальный объект-программа Application). Впервые введен в версии 5.

ValueListEditor - редактор строк, содержащих пары имя = значение. Пары такого типа широко используются в Windows, например, в файлах инициации, в системном реестре и т. п. Впервые введен в версии 6.

LabeledEdit - комбинация однострочного редактора и метки. Впервые введен в версии 6.

ColorBox - специальный вариант ComboBox для выбора одного из системных цветов. Впервые введен в версии 6.

Chart - диаграмма. Этот компонент облегчает создание специальных панелей для графического представления данных. Впервые введен в версии 3.

ActionManager - менеджер действий. Совместно с тремя следующими компонентами обеспечивает создание приложений, интерфейс которых (главное меню и инструментальные кнопки) может настраиваться пользователем. Впервые введен в версии 6.

ActionMainMenuBar - полоса меню, опции которого создаются с помощью компонента ActionManager. Впервые введен в версии 6.

ActionToolBar - полоса для размещения пиктографических кнопок, создаваемых с помощью компонента ActionManager. Впервые введен в версии 6.

CustomizeDig - диалог настройки. С помощью этого компонента пользователь может по своему вкусу настроить интерфейс работающей программы. Впервые введен в версии 6.

ТЕМА: Страница Dialogs

Страница Dialogs содержит компоненты, используемые для создания различных диалоговых окон, общепринятых в приложениях Windows. Диалоги используются для указания файлов или выбора установок. Применение поставляемых в составе Delphi диалоговых окон помогает сэкономить время на разработку и придать вашему приложению совместимость с принятыми в Windows нормами диалога.

OpenDialog - открыть. Реализует стандартное диалоговое окно “Открыть файл”.

SaveDialog - сохранить. Реализует стандартное диалоговое окно “Сохранить файл”.

OpenPictureDialog - открыть рисунок. Реализует специальное окно выбора графических файлов с возможностью предварительного просмотра рисунков.

SavePictureDialog - сохранить рисунок. Реализует специальное окно сохранения графических файлов с возможностью предварительного просмотра рисунков.

FontDialog - шрифт. Реализует стандартное диалоговое окно выбора шрифта.

ColorDialog - цвет. Реализует стандартное диалоговое окно выбора цвета.

PrintDialog - печать. Реализует стандартное диалоговое окно выбора параметров для печати документа.

PrinterSetupDialog - настройка принтера. Реализует стандартное диалоговое окно для настройки печатающего устройства.

FindDialog - поиск. Реализует стандартное диалоговое окно поиска текстового фрагмента.

ReplaceDialog - замена. Реализует стандартное диалоговое окно поиска и замены текстового фрагмента.

Компоненты OpenPictureDialog И SavePictureDialog введены в версии 3, остальные имеются в предыдущих версиях. Разумеется, интерфейс окон для Windows 16 (версия 1) отличается от интерфейса Windows 32.

ТЕМА: Массивы

Рассмотренные выше простые типы данных позволяют использовать в программе одиночные объекты - числа, символы, строки и т. п. В Object Pascal могут использоваться также объекты, содержащие множество однотипных элементов. Это *массивы* - формальное объединение нескольких однотипных объектов (чисел, символов, строк и т. п.), рассматриваемое как единое целое. К необходимости применения массивов мы приходим всякий раз, когда требуется связать и использовать целый ряд родственных величин. Например, результаты многократных замеров температуры воздуха в течение года удобно рассматривать как совокупность вещественных чисел, объединенных в один сложный объект - массив измерений.

При описании массива необходимо указать общее количество входящих в массив элементов и тип этих элементов. Например:

```
var
a : array [1..10] of Real;
b : array [0..50] of Char;
c : array [-3..4] of String;
```

Как видим, при описании массива используются зарезервированные слова array и of *(массив, из)*. За словом array в квадратных скобках указывается тип-диапазон, с помощью которого компилятор определяет общее количество элементов массива. Тип-диапазон (подробнее см. в гл. 7) задается левой и правой границами изменения индекса массива, так что

массив *A* состоит из 10 элементов, массив *B* - из 51, а массив C - из 8 элементов. За словом of указывается тип элементов, образующих массив.

Доступ к каждому элементу массива в программе осуществляется с помощью *индекса* - целого числа (точнее, выражения порядкового типа, см. гл. 7), служащего своеобразным именем элемента в массиве (если левая граница типа-диапазона равна 1, индекс элемента совпадает с его порядковым номером). При упоминании в программе любого элемента массива сразу за именем массива должен следовать индекс элемента в квадратных скобках, например:

```
var
a: array [1..10] of Integer;
b: array [0..40] of Char;
c: array [-2..2] of Boolean;
k: Integer;
begin
b[17] := 'F';
c[-2] := a[1] > 2;
for k := 1 to 10 do a[k] := 0;
end;
```

В правильно составленной программе индекс не должен выходить за пределы, определенные типом-диапазоном. Например, можно использовать элементы a [1], в[38], c[0], но нельзя a [0] илис[38] (определение массивов см. выше). Компилятор Object Pascal может[*Буквальный перевод array - боевой порядок, упорядоченная масса (войск). В компьютерной терминологии array переводится словом массив.*] контролировать использование индексов в программе как на этапе компиляции, так и на этапе прогона программы.

Учебная программа AVERAGE

Для иллюстрации приемов работы с массивами составим программу, которая создает массив случайных целых чисел, подсчитывает их среднее арифметическое, а также определяет и выводит на экран минимальное и максимальное из этих чисел.

```
procedure TfmExample.bbRunClick(Sender: TObject);
{Программа создает массив из N случайных целых чисел, равномерно распределенных в диапазоне от 0 до MAX_VALUE-1, подсчитывает среднее арифметическое этих чисел, а также минимальное и максимальное из них.}
const
  N = 1000; // Количество элементов массива
  MAX_VALUE = 100+1;// Диапазон значений случайных чисел var
  m: array [1..N] of Integer; // Массив чисел
  i: Integer;// Индекс массива
  max,min: Integer; // Максимальное и минимальное число
  sum: Real;// Сумма чисел
  begin
  // Наполняем массив случайными числами:
```

```
for i := 1 to N do
m[i] := Random(MAX_VALUE);
// Задаем начальные значения переменных:
sum : = m [ 1 ] ;
max : = m [ 1 ] ;
min := m[1] ;
// Цикл вычисления суммы всех случайных чисел и поиска
// минимального и максимального:
for i := 2 to N do
begin
sum := sum + m[i];
if m[i] < min then
min := m[i]
else if m[i] > max then
max := m[i] end;
// Вычисляем среднее значение и выводим результат:
IbOutput.Caption := 'Мин = '+IntToStr(min)+' Макс = '+ IntToStr(max)+'
Среднее = '+FloatToStr(sum/N) ;
end;
```

Для создания массива используется встроенная функция Random(Max), которая возвращает случайное целое число, равномерно распределенное в диапазоне от 0 до мах-1 (мах - параметр обращения).

ТЕМА: **Работа с графикой. Построение графиков.**

Для отображения графической информации в библиотеке Delphi предусмотрены компоненты, список которых дан в таблице

Компонент	**Страница**	**Описание**
Image (изображение)	Additional	Используется для отображения графики: пиктограмм, битовых матриц и метафайлов.
PaintBox (окно для рисования)	System	Используется для создания на форме некоторой области, в которой можно рисовать.
DrawGrid (таблица рисунков)	Additional	Используется для отображения в строках и столбцах нетекстовых данных.
Chart (диаграммы и графики)	Additional	Компонент принадлежит к семейству компонентов **TChart**, которые используются для создания диаграмм и графиков.

Кроме того, отображать и вводить графическую информацию можно на поверхности любого оконного компонента, имеющего свойство **Canvas** — канва.

Канва **Canvas** не является компонентом. Но поскольку многие компоненты, в частности, формы, имеют канву и канва предоставляет возможность выводить различную графическую информацию, то некоторые начальные сведения о канве все-таки целесообразно дать.

Канва представляет собой область компонента, на которой можно рисовать или отображать готовые изображения. Она содержит свойства и методы, существенно упрощающие графику Delphi. Все сложные взаимодействия с системой спрятаны для пользователя, так что рисовать в Delphi может человек, совершенно не искушенный в машинной графике.

Каждая точка канвы имеет координаты **X** и **Y**. Система координат канвы, как и везде в Delphi, имеет началом левый верхний угол канвы. Координата **X** возрастает при перемещении слева направо, а координата **Y** — при перемещении сверху вниз. Координаты измеряются в пикселях. Пиксель — это наименьший элемент поверхности рисунка, с которым можно манипулировать. Важнейшее свойство пикселя — его цвет.

Канва имеет свойство **Pixels**. Это свойство представляет собой двумерный массив, который отвечает за цвета канвы. Например, **Canvas.Pixels[10,20]** соответствует цвету пикселя, 10-го слева и 20-го сверху. С массивом пикселей можно обращаться как с любым свойством: изменять цвет, задавая пикселю новое значение, или определять его цвет по хранящемуся в нем значению. Например, **Canvas.Pixels[10,20] := 0** или **Canvas.Pixels[10,20] := clBlack** — это задание пикселю черного цвета.

Свойство **Pixels** можно использовать для рисования на канве. Давайте попробуем нарисовать по пикселям график синусоиды на канве формы. Для этого в обработчик события формы **OnPaint** (прорисовка) можно вставить следующий код:

```
procedure TForm1.FormPaint(Sender: TObject);
 var
  X,Y: real;      // координаты функции
  PX,PY: longint; // координаты пикселей
begin
 Color := clWhite;
 for PX := 0 to ClientWidth do
  begin
   {X — аргумент графика,
    соответствующий пикселю с координатой PX}
   X := PX*4*Pi/ClientWidth;
   Y:=Sin(X);
   {PY — координата пикселя,
    соответствующая координате Y}
   PY:=trunc(ClientHeight - (Y+1)*ClientHeight/2);
   {Устанавливается черный цвет выбранного
```

```
    пикселя (О яркости)}
    Canvas.Pixels [PX, PY] := 0;
  end;
end;
```

Выполните это тестовое приложение и вы увидите результат. График синусоиды получился, хотя и не очень хороший, т.к. разбивается на отдельные точки — пиксели.

Канва — объект класса **TCanvas** имеет множество методов, которые позволяют рисовать графики, линии, фигуры с помощью свойства **Pen** — перо. Это свойство является объектом, в свою очередь имеющим ряд свойств. Одно из них уже известное вам свойство **Color** — цвет, которым наносится рисунок. Второе свойство — **Width** (ширина линии). Ширина задается в пикселях. По умолчанию ширина равна 1.

Свойство **Style** определяет вид линии. Это свойство может принимать следующие значения:

psSolid	Сплошная линия
psDash	Штриховая линия
psDot	Пунктирная линия
psDashDot	Штрих-пунктирная линия
psDashDotDot	Линия, чередующая штрих и два пунктира
psClear	Отсутствие линии
psInsideFrame	Сплошная линия, но при Width > 1 допускающая цвета, отличные от палитры Windows

У канвы имеется свойство **PenPos** типа **TPoint**. Это свойство определяет в координатах канвы текущую позицию пера. Перемещение пера без прорисовки линии, т.е. изменение **PenPos**, производится методом канвы **MoveTo(X,Y)**. Здесь **(X, Y)** — координаты точки, в которую перемещается перо. Эта текущая точка становится исходной, от которой методом **LineTo(X,Y)** можно провести линию в точку с координатами **(X,Y)**. При этом текущая точка перемещается в конечную точку линии и новый вызов **LineTo** будет проводить точку из этой новой текущей точки.

Давайте попробуем нарисовать пером график синуса из предыдущего примера. В данном случае обработчик события формы **OnPaint** может иметь вид:

```
procedure TForm1.FormPaint(Sender: TObject);
var
 X,Y: real;      // координаты функции
 PX,PY: longint; // координаты пикселей
```

```
begin
 Color:=clWhite;
 Canvas.MoveTo(0,ClientHeight div 2);
 for PX:=0 to ClientWidth do
  begin
   {X - аргумент графика,
    соответствующий пикселю с координатой PX}
   X := PX*4*Pi/ClientWidth;
   Y := Sin(X);
   {PY — координата пикселя,
    соответствующая координате Y}
   PY := trunc(ClientHeight - (Y+1)*ClientHeight/2);
   {Проводится линия на  графике}
   Canvas.LineTo(PX,PY);
  end;
end;
```

Перо может рисовать не только прямые линии, но и фигуры. Полный список методов канвы, использующих перо, см. во встроенной справке Delphi. А пока в качестве примера приведем только один из них — **Ellipse**, который рисует эллипс или окружность. Он объявлен как

procedure Ellipse(X1, Y1, X2, Y2: Integer);

где параметры **X1, X2, Y1, Y2** определяют координаты прямоугольника, описывающего эллипс или окружность. Например, оператор

Canvas.Ellipse(10, 40, 20, 50);

нарисует окружность с диаметром 10 и с координатами центра (15, 45).

Фигуры в общем случае рисуются не пустыми, а закрашенными с помощью свойства канвы **Brush** — кисть. Свойство **Brush** является объектом, имеющим в свою очередь ряд свойств. Свойство **Color** определяет цвет заполнения. Свойство **Style** определяет шаблон заполнения (штриховку). По умолчанию значение **Style** равно **bsSolid**, что означает сплошное закрашивание цветом **Color**.

У пера **Pen** имеется еще одно свойство, которое мы пока не рассматривали. Это свойство — **Mode** (режим). По умолчанию значение **Mode = pmCopy**. Это означает, что линии проводятся цветом, заданным в свойстве **Color**. Но возможны и другие режимы, в которых учитывается не только цвет **Color**, но и цвет соответствующих пикселей фона. Наиболее интересным из этих режимов является режим **pmNotXor** — сложение с фоном по инверсному исключающему ИЛИ. Если задан этот режим, то повторное рисование той же фигуры на том же месте канвы убирает ранее нарисованное изображение и восстанавливает цвета пикселей, которые были до первого изображения фигуры.

Эту особенность режима **pmNotXor** можно использовать для создания простенькой анимации. Достаточно нарисовать нечто, затем стереть

нарисованное, перерисовать немного измененным — и рисунок будет представляться ожившим.

Попробуйте сделать сами простенькую мультипликацию — движущуюся окружность. Начните новое приложение и в раздел **implementation** вставьте объявление

var X,Y: integer;

Тем самым вы введете глобальные переменные **X** и **Y** — текущие координаты изображения.

В событие формы **OnPaint** вставьте операторы

```
Canvas.Brush.Color := clWhite;
Color := clWhite;
Canvas.Pen.Mode := pmNotXor;
```

Первый из этих операторов задает белый цвет кисти **Brush**. Значит ваша окружность будет закрашена внутри белым цветом. Второй оператор задает белый цвет фона поверхности формы. Третий оператор устанавливает режим пера **pmNotXor**, который позволит вам стирать прежнее изображение прежде, чем нарисовать новое.

Даже самая простая мультипликация нуждается в синхронизации. Иначе скорость движения будет определяться быстродействием компьютера. Поэтому перенесите на форму компонент **Timer** — таймер со страницы System. Можете посмотреть там его подробное описание. А пока задайте его свойство **Interval** равным, например, 30 (это время выдержки в миллисекундах, но реальное время выдержки будет больше — см. раздел 5.7) и установите свойство **Enabled** равным **false** (это означает, что таймер не будет запускаться автоматически в момент запуска приложения).

В обработчик события этого компонента **OnTimer** вставьте операторы

```
// Стирание прежнего изображения
Canvas.Ellipse(X-5, Y, X+5, Y-1Q);
Inc(X);
// Рисование  нового изображения
Canvas.Ellipse(X-5, Y, X+5, Y-10);
// Останов при достижении конца формы
if (X >= ClientWidth-20) then
  Timer1.Enabled := false;
```

Первый из этих операторов рисует окружность в том месте, где она была нарисована ранее, т.е. стирает прежнее изображение. Далее увеличивается на единицу функцией **Inc** текущая координата **X** и изображение окружности рисуется в новой позиции. Последний оператор останавливает изображение у края формы.

Теперь перенесите на форму кнопку **Button** и в обработчик щелчка на ней поместите операторы

```
X:=10;
Y:=100;
Canvas.Ellipse(X-5, Y, X+5, Y-10);
Timer1.Enabled:=true;
```

Первые два оператора задают начальные координаты окружности. Третий оператор рисует окружность в ее начальном положении, а четвертый — запускает таймер.

Оттранслируйте приложение, запустите его на выполнение, щелкните на кнопке. Вы увидите изображение окружности, перемещающееся по форме слева направо. А дальше уж подключите вашу фантазию и преобразуйте это не слишком интересное приложение во что-нибудь более увлекательное.

На канве можно отображать не только программно создаваемые изображения, но и изображения, хранящиеся в графических файлах. Только сама канва не имеет метода загрузки изображения из файла. Поэтому загружать файл надо в какой-нибудь другой графический объект, способный воспринимать информацию графических файлов. А затем переписывать изображение из этого объекта на канву с помощью метода канвы **Draw**. Его описание:

procedure Draw(X, Y: Integer; Graphic: TGraphic);

Здесь параметры **X** и **Y** определяют координаты левого верхнего угла размещения изображения на канве, а **Graphic** — объект, хранящий информацию. В качестве такого объекта может выступать, например, объект типа **TBitMap**, предназначенный для хранения битовых матриц. Давайте посмотрим, как все это выглядит на практике.

Откройте новое приложение, перенесите на форму компонент **OpenPictureDialog** со страницы Dialogs и кнопку **Button**. Разместите **OpenPictureDialog** в любом месте формы, так как этот компонент невизуальный, а кнопку разместите внизу формы. В обработчик щелчка на кнопке занесите код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  BitMap:TBitMap;
begin
 // Выбор пользователем графического файла
 if OpenPictureDialog1.Execute then
  begin
   // Создание объекта BitMap типа TBitMap
   BitMap:=TBitMap.Create;
   // Загрузка в BitMap выбранного графического файла
   BitMap.LoadFromFile(OpenPictureDialog1.FileName);
   // Перенос изображения на канву формы
   Canvas.Draw(10, 10, BitMap);
   // Уничтожение объекта BitMap
   BitMap.Free;
  end;
end;
```

Этот код создает временный объект типа **TBitMap** с именем **BitMap**. Затем вызывается диалог открытия графического файла **OpenPictureDialog1** и, если пользователь выбрал файл, то он загружается в **BitMap** методом

LoadFromFile. Затем методом **Draw** загруженное изображение копируется на канву в область, с координатами левого верхнего угла (10, 10). После этого временный объект **BitMap** уничтожается.

Запустите ваше приложение и щелкните на его кнопке. Вы увидите, что можете загрузить любой графический файл типа **.bmp** и он отобразится на канве формы.

Вы создали неплохое приложение для просмотра графических файлов. Но теперь давайте попробуем увидеть его крупный недостаток. Не закрывая своего приложения, перейдите в какую-нибудь другую программу, например, вернитесь в Delphi. Затем, ничего там не делая, опять перейдите в свое выполняющееся приложение. Если окно программы, в которую вы уходили, целиком перекрыло окно вашего приложения, то вернувшись в него вы увидите, что картинка в окне исчезла. Если же окно вашего приложения перекрывалось только частично, то вернувшись в свое приложение вы, возможно, увидите результат, подобный представленному на рис. 4.2 б.

Вы видите, что если окно какого-то другого приложения перекрывает на время окно вашего приложения, то изображение, нарисованное на канве формы, портится. Посмотрим, как можно устранить этот недостаток.

Если окно было перекрыто и изображение испортилось, операционная система сообщает приложению, что в окружении что-то изменилось и что приложение должно предпринять соответствующие действия. Как только требуется обновление окна, для него генерируется событие **OnPaint**. В обработчике этого события (в нашем случае события формы) нужно перерисовать изображение.

Перерисовка может производиться разными способами в зависимости от приложения. В нашем примере можно было бы вынести объявление переменной **BitMap** (оператор **var** BitMap: TBitMap) за пределы приведенной выше процедуры, т.е. сделать эту переменную глобальной, разместив непосредственно в разделе **implementation**. Оператор **BitMap.Free** можно было бы перенести в обработчик события формы **OnDestroy**, происходящего в момент закрытия приложения. Тогда в течение всего времени выполнения вашего приложения вы будете иметь копию картинки в компоненте **BitMap** и вам достаточно ввести в обработчик события **OnPaint** формы всего один оператор:

```
Canvas.Draw(10, 10, BitMap);
```

Сделайте это, и увидите, что изображение на форме не портится при любых перекрытиях окон.

Помимо рассмотренного метода **Draw** канва имеет еще метод копирования **CopyRect**:

```
procedure CopyRect(Dest: TRect; Canvas: TCanvas; Source: TRect);
```

Метод копирует указанную параметром **Source** область изображения в канве источника изображения **Canvas** в указанную параметром **Dest** область данной канвы. Тип **TRect**, характеризующий прямоугольные области **Source** и **Dest**. Например, оператор

```
Canvas.CopyRect(MyRect2, Bitmap.Canvas, MyRect1);
```

копирует на канву формы в область **MyRect2** изображение из области **MyRect1** канвы компонента **Bitmap**.

Копирование методом **CopyRect** производится в режиме, установленном свойством **CopyMode**. По умолчанию это свойство имеет значение **cmSrcCopy**, что означает просто замену изображения, содержащегося ранее в области **Dest**, на копируемое изображение. Другие возможные значения **CopyMode** позволяют комбинировать изображения.

***ТЕМА:* Диалоговые окна**

TOpenDialog и TSaveDialog - диалоги открытия и сохранения файлов

Эти компоненты имеют идентичные свойства и поэтому рассматриваются вместе. Пример окна TOpenDialog показан на рисунке.

Свойство FileName: string содержит маршрут поиска и выбранный файл при успешном завершении диалога. Программа может использовать это свойство для доступа к файлу с целью читать из него данные (TOpenDialog) или записывать в него (TSaveDialog). Замечу, что пользователь может ввести произвольное имя и, следовательно, указать несуществующий файл. Для записи это не имеет значения, но при чтении отсутствие файла может привести к краху программы. Чтобы избежать этого, можно проверить существование файла глобальной функцией FileExists, как это сделано в предыдущем примере, или использовать механизм обработки исключительных ситуаций.

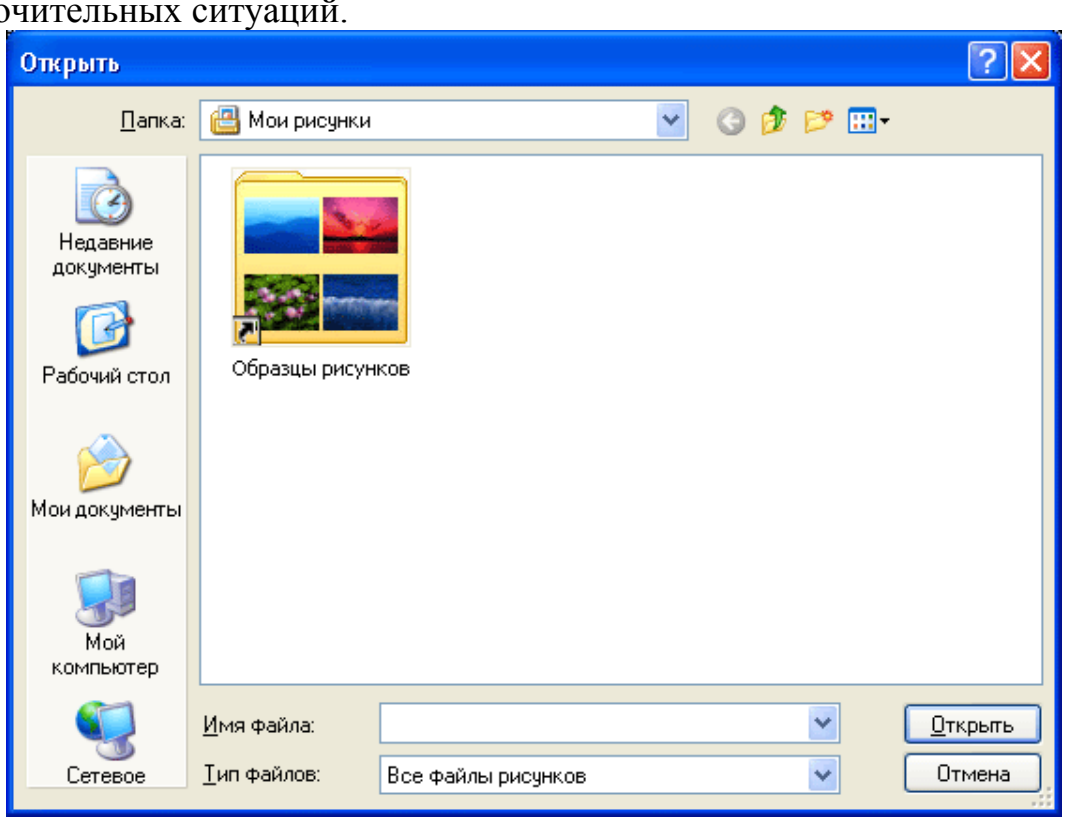


Рис. Стандартное окно TOpenDialog

Свойство Filter: string используется для фильтрации (отбора) файлов, показываемых в диалоговом окне. Это свойство можно устанавливать с помощью специального редактора на этапе конструирования формы или

программно, как это сделано в предыдущем примере. Для доступа к редактору достаточно щелкнуть по кнопке в строке Filter окна Инспектора объектов. При программном вводе фильтры задаются одной длинной строкой, в которой символы “|” служат для разделения фильтров друг от друга, а также для разделения описания фильтруемых файлов от соответствующей маски выбора. Например, оператор

OpenDialogI.Filter := 'Текстовые файлы|*.txt| Файлы Паскаля|*.pas';

задает две маски - для отбора файлов с расширениями pas и TXT.

Установить начальный каталог позволяет свойство InitialDir string. Например:

OpenDialogI.InitiaiDir := 'c:\program files\borland\delphi5\source';

С помощью свойства DefaultExt: String [3] формируется полное имя файла, если при ручном вводе пользователь не указал расширение. В этом случае к имени файла прибавляется разделительная точка и содержимое этого свойства.

В диалоговом окне для ручного ввода предусмотрен элемент TEdit, который при желании можно заменить на TCombovox. Для этого необходимо свойству FileEditStyle придать значение fsComboBox вместо умалчиваемого fsEdit. Если выбран комбинированный список, с ним можно связать протокол выбора имен. Для этого используется свойство HistoryList: TStrings, содержимое которого будет появляться в выпадающем списке. Этот список не пополняется автоматически, поэтому за его содержимым должна следить программа. Например:

if OpenDialogI.Execute **then**
begin
HistoryList.Add(OpenDialogI.FileName);
end;

Настройка диалога может варьироваться с помощью свойства

TOpenOption = (ofReadOnly, ofOverwritePrompt, ofHideReadOnly, ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMultiSelect, ofExtensionDifferent, ofPathMustExist, ofFileMustExist, ofCre-atePrompt, ofShareAware, ofNoReadOnlyReturn, ofNoTestFileCre-ate, ofNoNetworkButton, ofNoLongNames, ofOldStyleDialog, of-NoDereferenceLinks) ;

TOpenOptions = **set of** TOpenOption;

property Options: TOpenOptions;

Значения этого свойства имеют следующий смысл:

ofReadOnly	Устанавливает переключатель *Только для чтения*
ofOverwritePrompt	Требует согласия пользователя при записи в существующий файл
ofHideReadOnly	Прячет переключатель *Только для чтения*
OfNoChangeDir	Запрещает смену каталога
ofShowHelp	Включает в окно кнопку Help

ofNoValidate	Запрещает автоматическую проверку правильности набираемых в имени файла символов
OfAllowMultiSelect	Разрешает множественный выбор файлов
ofExtensionDifferent	При завершении диалога наличие этого значения в свойстве options говорит о том, что пользователь ввел расширение, отличающееся от умалчиваемого
ofPathMustExist	Разрешает указывать файлы только из существующих каталогов
ofFileMustExist	Разрешает указывать только существующие файлы
ofCreatePrompt	Требует подтверждения для создания несуществующего файла
ofShareAware	Разрешает выбирать файлы, используемые другими параллельно выполняемыми программами
OfNoReadOnlyReturn	Запрещает выбор файлов, имеющих атрибут *Только для чтения*
ofNoTestFileCreate	Запрещает проверку доступности сетевого или локального диска
OfNoNetworkButton	Запрещает вставку кнопки для создания сетевого диска
ofNoLongNames	Запрещает использование длинных имен файлов
ofOldStyleDialog	Создает диалог в стиле Windows 3-х

Если разрешен множественный выбор, доступ к выбранным именам можно получить в свойстве Files: strings.

TOpenPictureDialog и TSavePictureDialog - диалоги открытия и сохранения изображений

Специализированные диалоги для открытия и сохранения графических файлов TOpenPictureDialog И TSavePictureDialog отличаются от

TOpenDialog и TSaveDialog двумя обстоятельствами. Во-первых, в них предусмотрены стандартные фильтры для выбора графических файлов (с расширениями bmp, ico, wmf и emf) . Во-вторых, в окна диалога включены панели для предварительного просмотра выбираемого файла.

TFontDialog - диалог выбора шрифта.

Используется для настройки диалога. Значения свойств.

fdAnsiOnly	Показывает только шрифты с набором символов Windows
fdTrueTypeOnly	Показывает только TrueType-шрифты
fdEffects	Включает в окно переключатели *Подчеркнутый* и *Зачеркнутый,* а также список выбора цвета шрифта только моноширинные шрифты

fdFixedPitchOnly	Включает только моноширинные шрифты
fdForceFontExist	Предупреждает о выборе несуществующего шрифта
fdNoFaceSel	Запрещает выделение имени шрифта в момент открытия окна
fdNoOEMFonts	Запрещает выбор MS-DOS-шрифтов
fdNoSimulations	Исключает шрифты, которые синтезируются графическим интерфейсом Windows
fdNoSizeSel	Запрещает выделение размера шрифта в момент открытия окна
fdNoStyleSel	Запрещает выделение стиля шрифта в момент открытия окна
fdNoVectorFonts	Исключает векторные шрифты (шрифты для Windows версии 1.0; используются в плоттерах)
fdShowHelp	Включает в диалоговое окно кнопку Help шрифты, которые поддерживаются и экраном, и принтером
fdWysiwyg	Включает шрифты, которые поддерживаются и экраном, и принтером
fdLimitSize	Включает ограничения на размер шрифта, заданные свойствами MaxFontSize И MinFontSize
fdScalableOnly	Включает только масштабируемые шрифты (векторные и TrueType)
fdApplyButton	Включает в окно кнопку применить

Компонент **TPrintDialog** создает стандартное диалоговое окно для выбора параметров печати, показанное на рисунке.

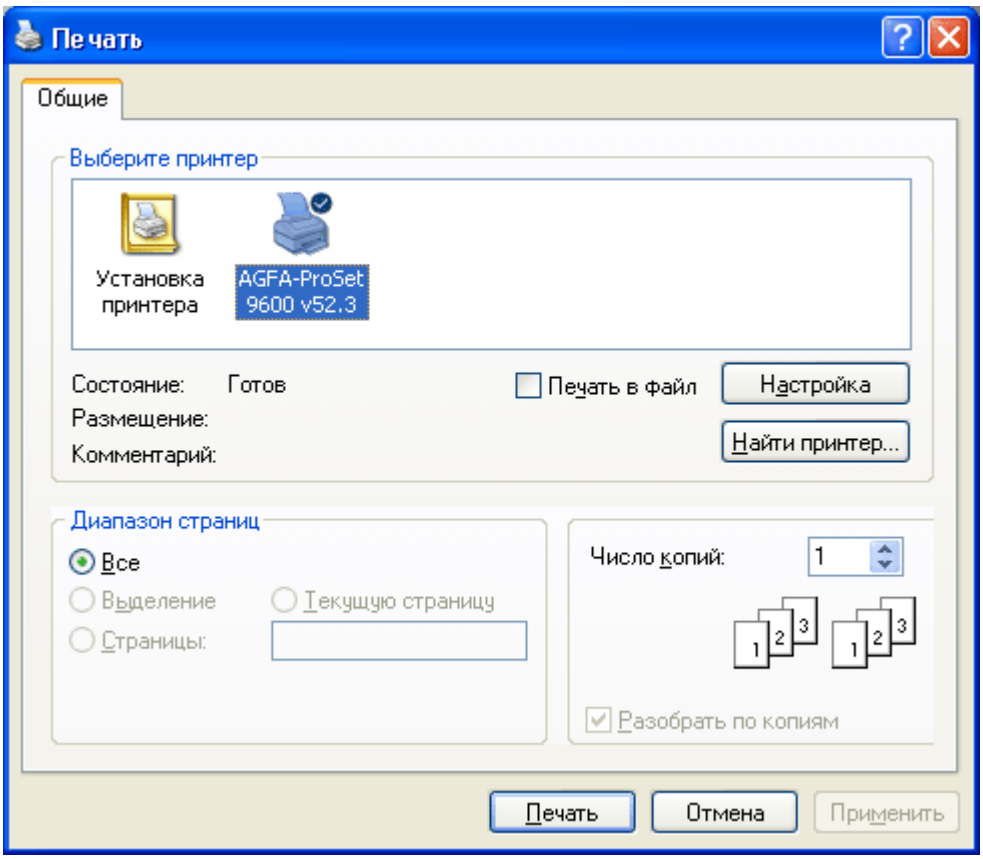


Рис. Стандартное окно компонента TPrintDialog

Свойства компонента:

property Collate: Boolean;	Если имеет значение True, окно показывается с выбранным переключателем разобрать (collate). Если этот переключатель выбран, печать нескольких копии документа будет идти по копиям: сначала первая копия, затем вторая и т. д., в противном случае - по страницам: сначала все копии первой страницы, затем второй и т. д
property Copies: Integer;	Определяет количество копии (0 - одна копия)
property FromPage: Integer	Определяет начальную страницу печати
property MaxPage: Integers;	Определяет верхнюю границу диапазона страниц для свойств FromPage, ToPage
property MinPage: Integer;	Определяет нижнюю границу диапазона страниц для свойств FromPage, ToPage
TPrintDialogOption = (poPrintToFile, poPageNums, poSelection, poWarning, poHelp, poDisablePrintToFile) ;	Определяет настройку окна: poPrintToFile - печатать в файл; poPrintToFile - разрешает выбор диапазона страниц; poSelection - разрешает печать выбранного текста; poWarning - предупреждать пользователя о неустановленном принтере; poHelp вставить в окно кнопку Help; poDisablePrintToFile -

TPrintDialogOptions = set of TPrintDialogOption; property Options: TPrintDialogOptions;	запрещает печать в файл
TPrintRange = (prAllPages, prSelection, prPageNums) ; property PrintRange: TPrintRange;	Определяет диапазон печатаемых страниц: prAllPages - все страницы; preelection - выделенный фрагмент текста; prPageNums - страницы по номерам
property PrintToFile: Boolean;	Содержит True, если пользователь выбрал печать в файл
property ToPage: Integer;	Определяет конечную страницу печати

ТЕМА: **Отладка программ**

В Delphi имеется мощный встроенный отладчик, значительно упрощающий отладку программ. Основными инструментами отладки являются точки контрольного останова и окно наблюдения за переменными.

Точки контрольного останова

Точка контрольного останова определяет оператор в программе, перед выполнением которого программа прервет свою работу и управление будет передано среде Delphi. Точка останова задается с помощью опции view | Debug windows | Breakponts.

Окно точек останова (рисунок) содержит список всех установленных в проекте точек, перед выполнением которых происходит прекращение работы программы и управление получает среда Delphi.

Для добавления новой точки следует щелкнуть по окну правой кнопкой мыши и выбрать опцию Add. В этом случае появляется окно, с помощью которого можно указать положение добавляемой точки: FileName - определяет имя файла;

Line number - номер строки от начала файла (в момент появления окна оно содержит файл и строку с текстовым курсором). В строке Condition можно указать условие останова в виде логического выражения (например, MyValue = Max-Value-12), а в строке Pass count - количество проходов программы через контрольную точку без прерывания вычислений.

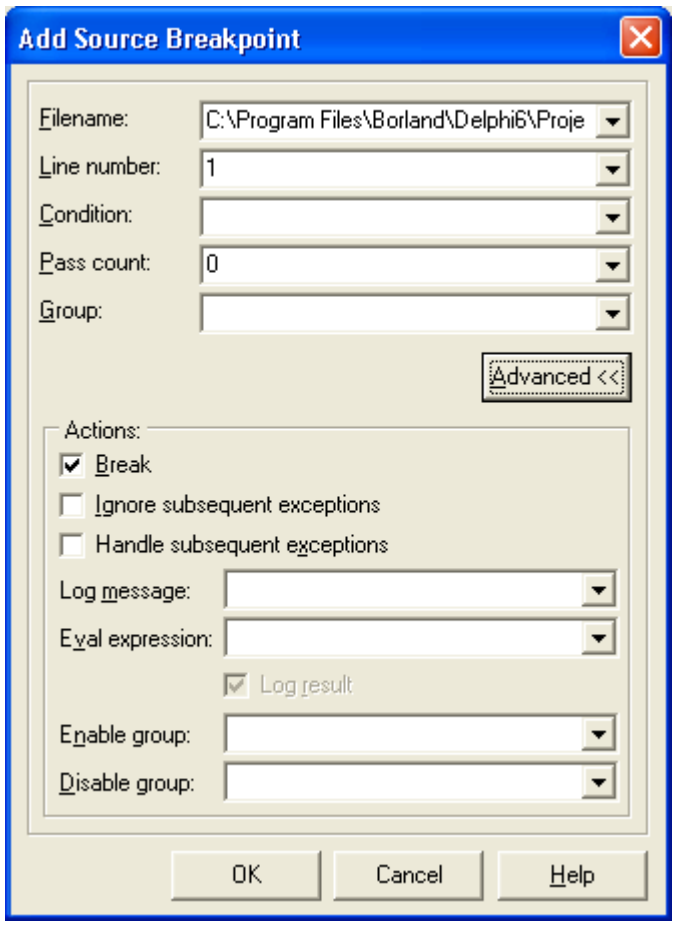


Рис. Окно точек останова (слева) и окно добавления новой точки (справа)

Принудительное прерывание работы программы

Если программа запущена из среды Delphi, ее работу можно прервать в любой момент G помощью клавиш Ctrl+F2, кнопки, опцией Run | program pause или, наконец, установив точку контрольного останова в той части программы, которая выполняется в данный момент или будет выполнена.

Трассировка программы

Перед исполнением оператора, в котором установлена точка контрольного останова, работа программы будет прервана, управление получит среда Delphi, а в окне наблюдения отразится текущее значение наблюдаемых переменных и/или выражений. Теперь программист может прослеживать работу программы по шагам с помощью клавиш F7 и F8 или инструментальных кнопок. При нажатии F8 будут выполнены запрограммированные в текущей строке действия, и работа программы прервется перед выполнением следующей строки текста программы. Замечу, что контрольная точка останова выделяется по умолчанию красным цветом, а текущая прослеживаемая строка - синим. Если программа остановлена в контрольной точке, т.е. когда текущая строка совпадает со строкой останова, строка выделяется красным цветом, Признаком текущей строки является особое выделение строки в служебной зоне слева в окне редактора.

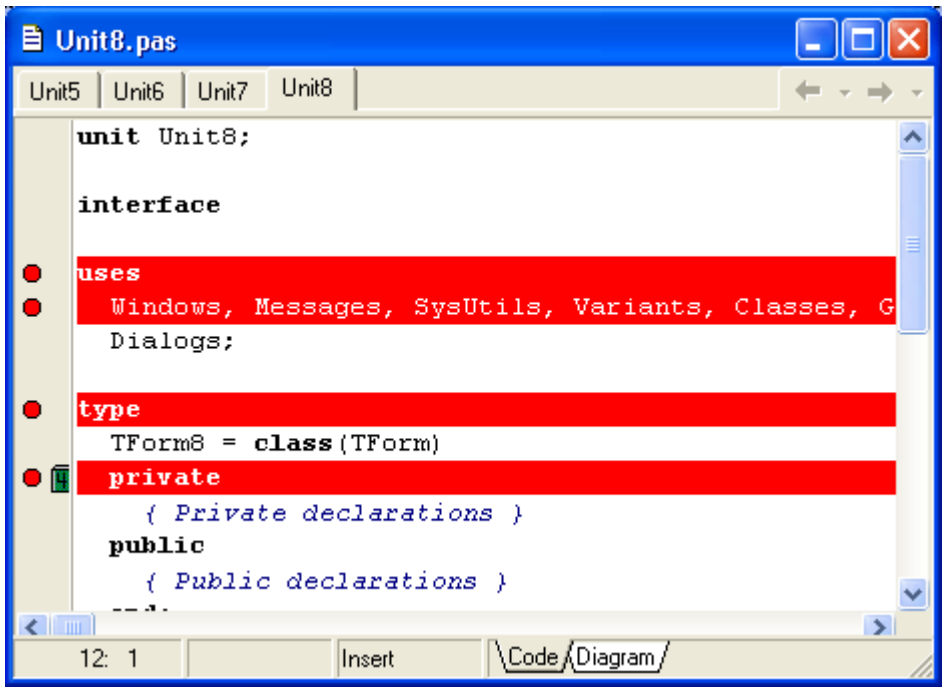


Рис. Фрагмент окна редактора в режиме отладки

Кстати, чтобы установить/снять точку контрольного останова, достаточно щелкнуть мышью по служебной зоне слева от нужной строки или установить в эту строку текстовый курсор и нажать F5.

При нажатии F7 среда выполняет те же действия, что и при нажатии F8, однако, если в текущей строке содержится вызов подпрограммы пользователя, программа прервет свою работу перед выполнением первого исполняемого оператора этой подпрограммы, т.е. клавиша F7 позволяет прослеживать работу вызываемых подпрограмм.

После трассировки нужного фрагмента программы можно продолжить нормальную ее работу, нажав клавишу F9.

Ведение протокола работы программы.

В ряде случаев бывает неудобно или невозможно пользоваться пошаговой отладкой программ. Если вы, например, установите точку останова в подпрограмме прорисовки сетки TDBGrid, программа после останова не сможет нормально продолжить свою работу, т. к. в этом случае она будет пытаться восстановить экран и вновь будет остановлена и т. д. В таких ситуациях вам могут помочь контрольные точки, которые не прерывают работу программы, а лишь помещают некоторую информацию в специальный файл трассировки. Для реализации такой точки раскройте окно RunlAdd Breakpoint | Source Breakpoint, уберите флажок в переключателе Break и напишите сообщение в строке Log message. Вы можете также в строке Eval expression указать некоторое выражение, которое будет вычислено и вместе с сообщением помещено в протокол работы программы. Этот протокол можно просмотреть в любой момент (в том числе и после завершения прогона программы) с помощью опции View] Debug Windows! Event Log.

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ И КОМПЛЕКТ ЗАДАНИЙ

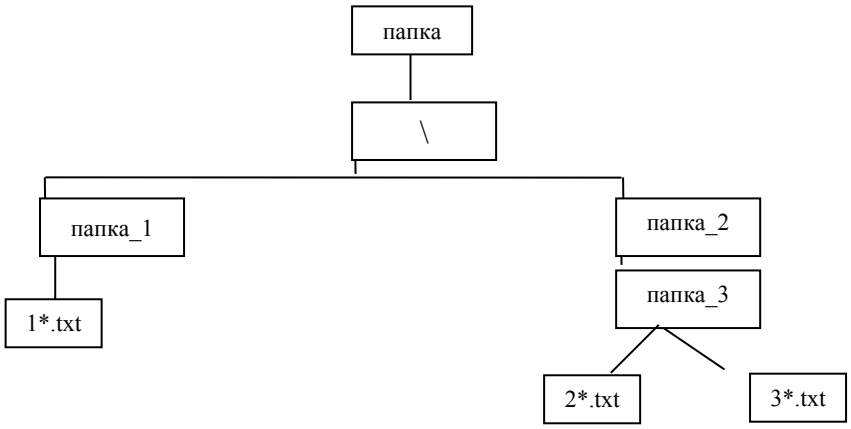
Лабораторный практикум предполагает самостоятельное выполнение студентом индивидуальных практических заданий по освоению теоретического материала. Текущий контроль выполнения работы осуществляется путем проверки преподавателем задания, выданного в соответствии с индивидуальным вариантом.

1 семестр

Лабораторная работа №1

Задание:

1. Создайте в соответствии с данной схемой каталоги, подкаталоги и текстовые фалы.



2. Введите в текстовые файлы текст. В файл с именем 1*.txt введите ваше ФИО, год рождении, группу специальность, на которой вы учитесь. В файл с именем 3*.txt введите ваше ФИО и специальность, на которую вы быть хотели поступить, но не поступили. Сравните содержимое этих файлов, результат - различия, если таковые имеются, сохранить в отдельном текстовом файле в корневом каталоге.
3. Войдите в корневой каталог и просмотрите содержимое данной папки.
4. введите произвольный текст в 2*.txt , сохраните, выйдете в корневой каталог и удалите файл 2*.txt , используя путь.
5. Сверьте время и дату на вашем компьютере, если время и дата не точны, внесите изменения.
6. Просмотрите содержимое файла 1*.txt, внесите изменения, добавив данные вашего домашнего адреса.
7. Удалите папку с именем «папка_1», предварительно очистив её.

Лабораторная работа №2

Тема: Изучение среды программирования Турбо Паскаль. Структура программы на Паскале. Линейные алгоритмы

```
Program my_prog1;   {Название программы. В принципе, не обязательно}
Uses crt, graph;    {Список подключаемых используемых модулей}
```

```
Type            {Далее следует список вводимых типов}
DigType = set of '0' .. '9' ;
StrType = string [40];
Const           {Затем – список вводимых констант}
N = 100;
EPS = 1e-9;
Var             {Список используемых переменных}
x,y:real ;
st :StrType;
Label           {Метки. Используются операторами безусловного перехода}
1b1, 1b2;
Function My_function(x,y: integer);{Функции, задаваемые пользователем}
Begin
…       {Тело функции}
End;
Procedure My_procedure;{Процедуры, задаваемые пользователем }
Begin
…       {Тело процедуры}
End;
Begin   {Начало исполняемой программы}
…       {Тело программы}
End.    {Конец исполняемой программы}
```

Пример 1 Простейшая программа, выводящая на экран текст «Я программирую на Турбо Паскале».

```
Program My_First_Program;
const
Text = 'Я программирую на Турбо Паскале';
begin
WriteLn(Text);
Readln;
end.
```

Пример 2 Программа, выводящая на экран звёздочками слово «Паскаль»

```
Program My_Second_Program;
begin
WriteLn('**** **** *****  *  * ****  ***   *   ');
WriteLn('*  * *  * *      * *  *  * *  *  *  *  ');
WriteLn('*  * **** ***    **   ****  *  *  ***  ');
WriteLn('*  * *  * *      * *  *  * *  *  *  *  ');
WriteLn('*  * *  * *****  *  * *  * *  *  *  ****');
Readln;
end.
```

Индивидуальные задания:

1. Напишите программу, выводящую на экран следующие фразы:
    1. «Программа выполнила недопустимую операцию и будет закрыта!»
    2. «Я люблю ФМиИ!»
    3. В центре экрана «Теперь питание компьютера можно отключить»
    4. «Идет форматирование диска: 98%»

5. «Файл не найден»
6. «Московское время: 12 часов.»
7. «Nokia – Connecting People!»
8. «Пока вы тут играетесь, ваш компьютер думает»
9. «В случае неполадки выдерни шнур и выбей стекло:-)»
10. «Техника в руках дикаря – это кусок железа»

Лабораторная работа №3

Тема: Программирование формул. Программы разветвляющей структуры. Табулирование функций.

Пример 1. Программа, вычисляющая в точках (1,1) (1, 2),…(10,10) значение функции z от переменных x и y по формуле:

$$z = \begin{cases} \frac{x}{y} + 2x & \text{если } x > y; \\ \frac{y}{x} + 2y & \text{если } x \le y; \end{cases}$$

```
Program Formula;
Uses Crt;
Var x,y,z: real;
Begin
For x:=1 to 10 do
Begin
  For y:=1 to 10 do
  Begin
    If (x>y) then
    z:=x/y+2*x
    else
    z:=y/x+2*y;
    Writeln('x='x,' y=',y,' z(x,y)=',z);
  End;
Readln;
End.
```

Задания: Протабулировать следующие функции на интервале [-10;10], исключая неопределённости.

№ варианта	Функция y=F(x)	№ варианта	Функция y=F(x)
1	$\frac{Sin(x)}{x}e^{-x}$	11	$Tg(3ArcTg(x))$
2	$x^x e^{-x}$	12	$e^{1-x^2}Ln(1+x^2)$
3	$\frac{ArcTg(x)}{x}e^{-x}$	13	$\frac{x(1+x)^3}{(1+x)^3-1}$
4	$x^3 e^{-Sin(\pi x)}$	14	$\frac{Cos(x)}{\pi/2-x}$

5	$x^{6,5}e^{-Cos(x)}$	15	$\frac{Sin(x)}{x}e^{-x}\left(1+x^3\right)$
6	$\frac{(1-x)\left(1-x^2\right)}{1-x^5}$	16	$\frac{1-e^x}{Sin(x)}$
7	$\frac{Ln(1+x)}{x}-\frac{Sin^2(x)}{x^2}$	17	$x^2Ln(x)e^{-x}$
8	$(1-x)^5e^{xSin(x)}$	18	$\frac{Ln(1+\pi x)}{x}e^{-x}$
9	$Cos(3ArcCos(x))$	19	$\frac{Tg(x)}{x}e^{-x}$
10	$Sin(3ArcSin(x))$	20	$x^xe^{-x^2}$

Лабораторная работа №4

Тема: Программы циклической структуры. Нахождение суммы ряда с точностью. Решение уравнений.

Пример 1.

Имеется убывающий ряд чисел: 1/2, 1/3, 1/4, 1/5,...

Найти сумму всех элементов ряда с точностью Z (т.е. если очередной элемент ряда стал меньше Z, то его суммировать уже не нужно)

```
program 10;
uses crt;
var
  z : Real;
  drob, b, c, a : Real;
  znamenat : Integer;
begin
ClrScr;
     Write('Z: ');
     ReadLn(z);
     drob := 0;
     znamenat := 1;
     a := 0;
      repeat
       a := 1 / (znamenat + 1);
       Inc(znamenat);
       drob := drob + a
      until a <= z;
     WriteLn('Ответ: ', drob:10:2);
     WriteLn;
     Write('Press Enter');
     ReadLn
end.
```

Пример 2. Найти корень уравнения $x^2 - 15x+50 = 0$ на интервале [0;7] методом деления отрезка пополам.

```
Program BiSect;
Uses Crt;
Var
  Iter:Integer;
  A, B, c, Fa, Fc, X, Eps: Real;
Begin
Eps:=0.0001
A:=0;
B:=7;
Fa:=A*A-15*A+50; {Вычислить значение функции в левой границе отрезка}
Repeat {Начало цикла деления отрезка пополам. Цикл с постусловием}
   c:=(A+B)/2; {Вычислить середину интервала}
   Fc:=c*c-15*c+50; {Вычислить значение функции в середине отрезка}
   If Fa*Fc<=0 {Если значения функции в точках a и c имеют разные знаки}
   then B:=c {То конец отрезка перенести в середину}
   else A:=c {В противном случае начало отрезка перенести в середину}
until (B-A)<Eps;
  X:=(A+B)/2 {Как результ – середина последнего, самого короткого отрезка}
Writeln(X);
Readln;
End.
```

Задания:

Найти суммы следующих рядов:

1. $\sum_{n=1}^{30} \frac{n!}{n^n}$;
2. $\sum_{n=1}^{50} \frac{2n}{n+2n}$
3. $\sum_{n=1}^{20} \frac{1}{n}$;
4. $\sum_{n=1}^{30} n^3 + n^2 + n + 1$
5. $\sum_{n=1}^{50} \sqrt{2n+1}$;
6. $\sum_{n=1}^{50} 2^n$
7. $\sum_{n=1}^{30} n^{2+n}$;
8. $\sum_{n=1}^{80} \frac{n^8}{n^7}$
9. $\sum_{n=1}^{50} \frac{n}{\sqrt{n+1}}$;
10. $\sum_{n=1}^{50} \frac{\sqrt{n+1}}{n}$

Решить численно следующие уравнения:

Методом половинного деления:

1. $Tg(x) = 1/x$
2. $Ln(x) = 1/x$
3. $e^{-x} = x$

Методом хорд:

4. $Ln(x) = 1/x^2$

5. $e^{-x^2} = x$

6. $x + 3 = x^3$

7. $x - x^3 + 1 = 0$

Методом Ньютона:

8. $Sin(x) = x/3$

9. $Cos(x) = 1/x$

10. $Cos(x) = x$

Лабораторная работа №5

Тема: Вложенные циклы

Пример 1. Найти все четырехзначные числа, у которых все цифры различны.

```
 program razn;
var a1, a2, a3, a4: byte;
begin
 for a1: =1 to 9 do
  for a2:= 0 to 9 do
   for a3:= 0 to 9 do
    for a4:= 0 to 9 do
     if (a1<>a2) and (a1<>a3) and (a1<>a4) and (a2<>a3) and (a2<>a4) and (a3<>a4) then
       write(a1,a2,a3,a4,'   ');
 readln;
end.
```

Пример 2. Найти в промежутке от 1 до 1000 числа, у которых пять делителей.

```
program 5div;
uses crt;
var s,del,n:integer;
begin
  clrscr;
    for n:=1 to 1000 do
      begin
        s:=0;
           for del:=1 to n do
             if n mod del = 0 then inc(s);
          if s=5 then write(n,'  ');
      end;
  readln;
end.
```

Задания.

1. Натуральное число из n цифр является число Армстронга, если сумма его цифр возведенных в n-ую степень равна самому числу. Получите все эти числа состоящие из трех и четырех цифр.
2. Найти в промежутке от 0 до 100 все простые числа.
3. Найти все числа, делящие числа от 50 до 100 нацело.
5. Написать программу, разлагающую любое число на простые множители.

6. Написать программу, складывающую остатки от деления чисел от 1 до N на числа из интервала от 1 до N-1 (N>2).
7. Из двух последовательностей натуральных чисел составить множество попарных произведений, образованных простыми числами.
8. Из двух последовательностей натуральных чисел составить множество попарных произведений, оканчивающихся на цифру k,
9. Составить программу, преобразующую последовательность натуральных чисел от n до n+k следующим образом: первый член последовательности умножается на 2, второй – на 2^2, третий – на 2^3 и т.д.
10. Составить программу, преобразующую последовательность натуральных чисел от 1 до n следующим образом: первый член последовательности умножается на -1, второй – на 2^3, третий – на -3^4 и т.д.

Лабораторная работа №6
Тема: Вычисление интеграла.
Пример 1. Численное интегрирование функции $y = x^2 – 10x + 25$ методом прямоугольников:

```
program integral;
var
n,i : integer;
a,b,shag,sum,itog : real;
function F(x:real):real;
begin
F:=x*x-10*x+25
end;
BEGIN
write('Начало интегрирования a = '); readln(a);
write('Конец интегрирования b = '); readln(b);
write('Количество разбиений интервала n = '); readln(n);
shag:=(b-a)/n;
sum:=0;
for i:=1 to n-1 do
sum := sum + F(shag*i+a);
sum := sum + (F(a)+F(b))/2;
itog:=(b-a)/n * sum;
writeLn('Интеграл = ', itog:0:5)
END.
```

Пример 2 Численное интегрирование той же функции методом Симпсона (четное количество разбиений интервала интегрирования):

```
program Simpson;
function F(x:Real):Real;
begin
F:= x*x-10*x+25;
end;
var a,b,h,x :real;
```

```
n,i :integer;
integ :real;
begin
write('Введите начало интегрирования a='); readln(a);
write('Введите конец интегрирования b='); readln(b);
write('Введите количество разбиений интервала (четное число) n=');
readln(n);
if (n mod 2)>0 then
begin
n:=n+1;
writeln('Число n было введено нечетное, оно было заменено на n=',n);
end;
h:=(b-a)/n;
integ:=F(a)+F(b)+4*F(a+h);
for i:=1 to (n div 2)-1 do
begin
x:=a+2*h*i;
integ:=integ+2*F(x)+4*F(x+h);
end;
integ:=h*integ/3;
writeln('Интеграл = ',integ);
end.
```

Задания:

Найти численно значения следующих интегралов:

Методом левых прямоугольников:

1. $\int_{0}^{10} \frac{x^2}{2} dx$
2. $\int_{-5}^{5} \left(\frac{(x-2)^3}{2} + 1 \right) dx$
3. $\int_{-\pi}^{\pi} \frac{\ln x}{x} dx$
4. $\int_{-\pi}^{2\pi} \frac{\sin x}{x} dx$
5. $\int_{-\pi}^{2\pi} \frac{\sin x}{x} dx$

Методом правых прямоугольников:

6. $\int_{0}^{2\pi} \frac{dx}{\cos x + 1}$
7. $\int_{0}^{2\pi} x^2 \cos x dx$

8. $\int_0^{2\pi} x^2(\cos x + 1)dx$

9. $\int_0^5 x^3\sqrt{25 - x^2}\,dx$

10. $\int_0^{0.75} \frac{dx}{(x+1)\sqrt{x^2+1}}$

Методом трапеций:

11. $\int_{-1}^{1} \frac{xdx}{\sqrt{5-4x}}$

12. $\int_{-1}^{1} \frac{1+x^2}{1+x^4}dx$

13. $\int_0^{100\pi} \sqrt{1-\cos 2x}dx$

14. $\int_0^{2\pi} \ln\sin x dx$

15. $\int_{1/e}^{e} |\ln x|dx$

Методом Симпсона:

16. $\int_0^{\ln 2} \sqrt{e^x - 1}dx$

17. $\int_0^{10} \frac{dx}{x^2+1}$

18. $\int_0^5 \frac{xdx}{\sqrt{x^2+1}}$

19. $\int_0^{\ln 2} \frac{dx}{\sqrt{e^x - 1}}$

20. $\int_{1/2}^{2} \left(1 + x - \frac{1}{x}\right)e^{x+\frac{1}{x}}dx$

Лабораторная работа №7

Тема: Работа с массивами

```
Program mid;
{Программа вычисляет среднее арифметическое значений
элементов массива}
Const nn = 10;
Var
        a: Array[1..nn] Of Integer;
        I, n: Integer;
        S: Real;
Begin
{Ввод количества элементов массива}
```

```
        Write('Количество элементов?'); Readln(n);
{Ввод элементов массива}
        For i := 1 To n Do
        Begin
                Write('Введите a[' , i, ']'); Readln(a[i]);
        End;
{Вычисление суммы элементов массива}
        S := 0;
        For i := 1 To n Do S := S + a[i];
{Вывод результата на экран}
        Writeln('Результат: ', (S/n):5:3);
End.
```

Задание:

Составить схему алгоритма, написать и отладить программу в соответствии с вариантом.

№ варианта	Задача
1	1. Вычислить длину вектора X размером 4. Длина вектора вычисляется по формуле $L = \sqrt{X_1^2 + ... + X_4^2}$. 2. Задан массив $X_1, X_2, \ldots, X_m$. Вывести на экран номера элементов равных 0.
2	1. Задан массив $b_1, b_2, b_3, \ldots b_n$. Все элементы, равные 7 заменить единицей и подсчитать количество таких элементов. 2. Осуществить центрирование массива: от каждого из заданных m чисел $X_1, X_2, \ldots, X_m$ отнять их среднее арифметическое и результаты разместить на месте исходных данных.
3	1. Вычислить среднее значение $X_{ср}$ и дисперсию d_x для заданного массива наблюдений $X_{ср} = \frac{1}{k}\sum_{i=1}^{k} X_i;\ d_x = \frac{1}{k-1}\sum_{i=1}^{k} (X_i - X_{ср})^2;$ 2. Задан массив $b_1, b_2, b_3, \ldots b_n$. Расположить элементы массива в порядке убывания. Вывести на экран исходный и полученный в результате сортировки массивы.
4	1. Вычислить скалярное произведение двух векторов. Скалярное произведение вычисляется по формуле $C = \sum_{i=1}^{k} X_i Y_i;$ 2. Задан массив $X_1, X_2, \ldots, X_m$. Найти номер минимального элемента массива.
5	1. Решить уравнение ax = b для пяти пар значений a и b, заданных в виде двух массивов. Результат поместить в массив X. 2. Задан массив $a_1, a_2, a_3, \ldots a_n$. Найти количество элементов, значения которых находятся в интервале [1…5].

Дополнительные задания:

1. Задан массив $X_1, X_2, \ldots, X_m$. Исключить из массива максимальный элемент.
2. Записать в массив C по возрастанию элементы массивов a1, a2, a3, … an и b1, b2, b3, … bn. Элементы массивов A и B расположены в случайном порядке.

3. В массиве a1, a2, a3, … an найти номера локальных максимумов (ai-1< ai,> ai+1) и расположить их в порядке убывания (локальные максимумы).
4. Задан массив a1, a2, a3, … an . Первую половину массива упорядочить по возрастанию. Во второй половине массива найти минимальный элемент.
5. Задан массив a1, a2, a3, … an . Переписать в массив b все элементы, следующие за максимальным элементом массива

Лабораторная работа №8

Тема: Сортировка массивов

Пример 1.

Сортировка массива методом пузырька.

```
program pyz;
const n = 20;
var V:array[1..n] of integer;
   i,j,tmp:integer;
begin
Randomize;
 for i:=1 to n do
  V[i]:= Random(100);
Writeln('Исходный массив: ');
 for i:=1 to n do Write(V[i],' ');

 for i:=1 to n-1 do
  for j:=i+1 to n do
   if V[i] >= V[j] then begin
                        tmp:= V[i];
                        V[i]:= V[j];
                                V[j]:= tmp;
                        end;
Writeln;
Writeln('Упорядоченный по неубыванию массив: ');
 for i:=1 to n do Write(V[i],' ');
Readln;
End.
```

Пример 2. Сортировка массива по возрастанию методом прямого выбора

```
program sortarr;
const
    SIZE=5;
var
    a:array[1..SIZE] of integer;
    i:integer;{№ эл-та, от которого ведется поиск минимального}
    min:integer; {№ мин. эл-та в части массива от i до верхней границы}
    j:integer; {№ эл-та, сравниваемого с минимальным}
    buf:integer; {буфер, используемый при обмене эл-тов массива}
    k:integer;
begin
    writeln('Сортировка массива.');
    write('Введите',SIZE:3,' целых в одной строке ');
    writeln('через пробел и нажмите <Enter>');
    for k:=1 to SIZE do read(a[k]);
```

```
    writeln('Сортировка');
    for i:=1 to SIZE-1 do
    begin
        { поиск минимального эл-та
        в части массива от a[i] до a[SIZE]}
        min:=i;
        for j:=i+1 to SIZE do begin
            if a[j]<a[min] then min:=j;
            { поменяем местами a[min] и a[i] }
            buf:=a[i];
            a[i]:=a[min];
            a[min]:=buf;
           {выведем массив }
        for k:=1 to SIZE do write(a[k],' ');
        writeln;
              end;
    end;
    writeln('Массив отсортирован.');
end.
```

Задания:

Дан случайный массив из натуральных чисел.Составить программу, которая выполняет сортировку его элементов следующим образом:

1. Сначала идут элементы, кратные числу K (дают 0 в остатке от деления на K), затем те, которые при делении на K дают в остатке 1, и т.д. Последнюю группу составляют числа, остаток от деления на K которых равен K-1.
2. По убыванию, методом пузырька.
3. По убыванию, методом прямого выбора.
4. Половина массива по возрастанию, половина – по убыванию.
5. Сначала все нечётные – по возрастанию, затем все нечётные по возрастанию.
6. Сначала все простые числа – по возрастанию, затем все остальные – по убыванию.
7. Сначала по возрастанию - все кратные 2, затем, все кратные 3, затем, все кратные 5, затем, все кратные 7, затем все остальные по убыванию.
8. Все элементы на нечётных местах – по возрастанию, остальные – по убыванию.
9. Сначала все элементы, заканчивающиеся цифрой 1, затем – цифрой 2, затем – цифрой 3 и т.д.
10. Сначала все элементы, начинающиеся цифрой 1, затем – цифрой 2, затем – цифрой 3 и т.д.

Лабораторная работа №9

Тема: Обработка матриц

Пример. Дана таблица целых чисел, содержащая 10 строк и 20 столбцов. Сосчитать сумму всех чисел в таблице.

```
Program MASSIV;
VAR
```

```
i,j : Integer;
SUMMA: Integer;
A : Array [1..10, 1..20] of Integer:
BEGIN
SUMMA:= 0;
Writeln(‘ Введите массив A');
For i:=l to 10 do
For j:=l to 20 do
Begin
Write(‘A[‘,,’,’,j,’]=’);
Readln (A [i,j]);
SUMMA:= SUMMA+A[i,j];
End;
Writeln('Сумма =’, SUMMA);
END.
```

Пример 2. Задан двумерный целочисленный массив, программа находит в нём наименьший и наибольший элементы.

```
program minImax;
uses crt;
const
    N = 9;
type Mas = array[1..N,1..N] of integer;
var
  M: Mas;
  i,j: integer;
  min,max,jmin,imax: integer;
Begin
clrscr;
Writeln;
Randomize;
TextAttr:=15;
for i:=1 to N do begin        {инициализация и вывод матрицы}
 for j:=1 to N do begin
 M[i,j]:=Random(10);
 Write(' ',M[i,j]:3);
               end;
 Writeln;
             end;
max:=M[1,1];imax:=1;
min:=M[1,1];jmin:=1;

for i:=1 to N do begin        {поиск максимума и минимума в матрице}
 for j:=1 to N do begin
  if M[i,j] > max then begin
                  max:=M[i,j];imax:=i;
                  end;
  if M[i,j] < min then begin
                  min:=M[i,j];jmin:=j;
                  end;
  end;
 end;
Writeln;
```

```
Writeln('max= ',max,' в строке ',imax);
Writeln('min= ',min,' в столбце ',jmin);
Readln;
End.
```

Задания:

1. Среди тех строк целочисленной матрицы, которые содержат только нечётные элементы, найти строку с максимальной суммой модулей элементов.
2. Дана произвольная числовая матрица. Заменить отрицательные элементы нулями, положительные единицами.
3. Сформировать квадратную матрицу

n 0 0 0...0 0 0
0 n-1 0 0...0 0 0
0 0 n-2 0...0 0 0
. . . . . .
0 0 0 0...0 2 0
0 0 0 0...0 0 1

4. Задана квадратная матрица. Переставить строку с максимальным элементом на главной диагонали со строкой с заданным номером m.
5. Составить матрицу вида:

11111
01110
00100
01110
11111

6. Составить диагонально - единичную матрицу размерностью 20 на 20.
7. Задана случайная числовая квадратная матрица. Заменить все элементы главной диагонали нулями.
8. Задана случайная числовая квадратная матрица. Заменить все элементы побочной диагонали единицами.
9. Составить матрицу 50 на 50, размещая в первой строке одни единицы, во второй – одни двойки, в третьей – одни тройки и т.д.
10. Составить программу умножения двух квадратных соразмерных матриц.

2 семестр

Лабораторная работа №1

Тема: Рекурсия

Пример 1. Программа, вычисляющая факториал числа.

```
program Factorial;
var n:integer;
function Factor(n:integer):real;
 var v:real;
 Begin
 if (n=0) or (n=1) then Factor:= 1 else Factor:= n*Factor(n-1);
 end;
```

```
begin
 Write('Введите число(0..33): ');
 Readln(n);
 Write('Факториал этого числа равен: ', Factor(n):11:0);
 Readln;
End.
```

Задания:

Написать программу, вычисляющую любой номер числа последовательности:

1. $a_{n+1} = \frac{a_{n-1}}{a_n}$
2. $F_n = F_{n-1} + F_{n-2}$ (Числа Фибоначчи)
3. $a_{n+1} = a_{n-1}a_n$
4. $a_{n+1} = 3a_{n-2} + 2a_{n-1} + a_n$
5. $a_{n+1} = a_{n-2} + 2a_{n-1} + 3a_n$
6. $a_{n+1} = a_{n-2} + 2a_{n-1} + 3a_n$
7. $a_{n+1} = a_{n-1}^2 + 2a_{n-1}a_n + a_n^2$
8. $a_{n+1} = a_{n-1}^2 - \frac{4a_{n-1}}{a_n} + a_n^2$
9. $a_{n+1} = \frac{a_n}{a_{n-1}} + \frac{a_{n-1}}{a_{n-2}} + \frac{a_{n-2}}{a_{n-3}}$
10. $a_{n+1} = a_n a_{n-1} + \frac{a_{n-1}}{a_{n-2}} + a_{n-2}a_{n-3}$

Лабораторная работа №2

Тема: Решение задач с использованием подпрограмм

Пример 1. Пример использования процедуры

```
program procline;
Procedure Line30; { Процедура вывода строки из 30 черточек }
    var
        i:integer;
    begin
        for i:=1 to 30 do write('-');
        writeln;
    end;
{основная программа }
var
    x:integer;
    y:real;
begin
    writeln('Таблица логарифмов чисел от 1 до 10');
    Line30;     { нарисовать линию }
    for x:=1 to 10 do begin
        y:=ln(x);
```

```
        { функция вычисления логарифма x }
        writeln(x:3,y:8:3);
    end;
    Line30;     { нарисовать линию }
end.
```

Пример 2. Пример использования функции.

```
Function cubrt(x:real):real; {Функция "Кубический корень"}
var
    pr:real;     {приближенное значение кубического корня}
begin
    pr:=sqrt(x); { первое приближение }
    {в качестве второго приближения выбираем (x:pr):pr }
    while abs(pr-x/(pr*pr))>0.001 do
    begin
        {новое приближение - среднее арифметическое
         удвоенного приближения на прошлом шаге и текущего}
        pr:=(2*pr+x/(pr*pr))/3;
    end;
    cubrt:=pr;
end;
{Проверим, как работает cubrt }
begin
         writeln(cubrt(3));
         readln;
end.
```

Задания:

Написать следующие функции для программ:

1. Функция тангенса
2. Функция котангенса
3. Функция секанса
4. Функция косеканса
5. Функция гиперболического синуса
6. Функция гиперболического косинуса
7. Функция определителя для матрицы 3 на 3
8. Функция корня четвертой степени из числа
9. Функция возведения в пятую степень
10. Функция Sign (возвращает 1 для положительных и –1 для отрицательных чисел.

Написать следующие процедуры для программ:

1. Процедура сортировки массива методом пузырька
2. Процедура сортировки массива методом прямого выбора
3. Процедура решения произвольного квадратного уравнения
4. Процедура вычисления интеграла по формуле прямоугольников
5. Процедура вычисления интеграла по формуле Симпсона
6. Процедура вычисления интеграла по формуле трапеций
7. Процедура решения произвольного уравнения методом половинного деления

8. Процедура нахождения суммы произвольного числового ряда
9. Процедура задания произвольного массива из 100 элементов
10. Процедура задания произвольной матрицы 20 на 20

Лабораторная работа №3
Тема: Строки
Пример 1. Работа со строками

```
var
x : Real;
y : Integer;
st,st1: String;
begin
st := concat('12','345'); {строка st содержит 12345}
st1 := copy(st,3,Length(st)-2); {st1 содержит 345}
insert('-',st1,2); {строка st1 содержит 3-45}
delete(st,pos('2',at),3); {строка st содержит 15}
str(pi:6:2,st); {строка st содержит 3.14}
val("3,1415' ,x,y) ; {y содержит 2, x остался без изменения}
end.
```

Задания:
1. Дан текст на русском языке. Напечатать в алфавитном порядке все глухие согласные буквы, которые входят в каждое нечётное слово и не входят хотя бы одно чётное слово.
2. Подсчитать в строке число букв A и B, если букв A больше, чем B, то удалить в строке все символы B.
3. Подсчитать количество слов в строке.
4. Дана символьная строка, заканчивается точкой. Найти длину самого длинного и самого короткого слова.
5. Найти в строке слова, начинающиеся и заканчивающиеся на одну и ту же букву.
6. Вводится последовательность латинских букв, признаком конца которой является пробел. Напечатать эту последовательность, упорядоченную по алфавиту.
7. Программа должна считывать символы и затем запрашивать строку на ввод. Проверить, можно ли из введенных символов составить данную строку.
8. Ввести две строки, изъять из первой строки все слова, которые встречаются во второй.
9. Написать процедуру, заменяющую звёздочками все слова, состоящие из трёх букв.
10. Написать программу, проверяющую, является ли данное слово палиндромом (пишется одинаково в обе стороны).

Лабораторная работа №4
Тема: Множества
Пример 1. Работа с множествами

```
Program Primer_numbers_detect;
{Выделение всех простых чисел из первых N целых}
```

```
const
N = 255; {Количество элементов исходного множества}
type
SetOfNumber = set of 1..N;
var
n1,next,i : Word; {Вспомогательные переменные}
BeginSet, {Исходное множество}
PrimerSet : SetOfNumber; {Множество простых чисел} .
begin
BeginSet := [2. .N] ; {Создаем исходное множество}
PrimerSet:= [1]; {Первое простое число}
next:= 2; {Следующее простое число}
while BeginSet <> [] do {Начало основного цикла}
begin
n1 := next;{n1-число,кратное очередному простому (next)}
{Цикл удаления из исходного множества непростых чисел:}
while n1 <= N do
begin
Exclude(BeginSet,nl);
n1 := n1+next {Следующее кратное}
end; {Конец цикла удаления}
Include(PrimerSet,next);
{Получаем следующее простое, которое есть первое невычеркнутое из исходного
множества}
repeat
inc(next)
until (next in BeginSet) or (next > N)
end; {Конец основного цикла}
{Выводим результат:}
for i := 1 to N do
if i in PrimerSet then Write(i:8);
WriteLn
end.
```

Задания:

1. Составить программу, используя величины множественного типа.
Есть список имен кукол: "Барби", "Люси", "Катерина", "Светлана", "Марина", "Анжела". Имеется информация о том, куклы с какими именами есть у N девочек. Составить программу, которая выводит список кукол:
а) имеющихся у каждой из девочек;
б) имеющихся хотя бы у одной из девочек;
в) которых нет ни у одной из девочек.
2. Дан текст на русском языке. Напечатать в алфавитном порядке все глухие согласные буквы, которые не входят хотя бы в одно слово.
3. Имеется список класса (все имена различны). Определить, есть ли в классе человек, который побывал в гостях у всех. Для каждого ученика составить множество побывавших у него в гостях друзей, сам ученик в это множество не входит.
4. Напечатать в возрастающем порядке все цифры, не входящие в запись данного натурального числа.

5. В алфавитном порядке вывести все звонкие согласные, которые входят в каждое нечетное слово и не входят ни в одно четное слово.
6. Написать программу, производящую фонетический анализ слова (по буквам: согласная/гласная, звонкая/глухая, звучит твёрдо/мягко).
7. Написать программу, удаляющую из строки все гласные буквы.
8. Написать программу, удаляющую из строки все согласные буквы.
9. Написать программу, определяющую, является ли человек с данным именем клиентом фирмы с фиксированным списком клиентов
10. Написать программу, выводящую фамилию человека после ввода его имени (все иена и фамилии различны).

Лабораторная работа №5
Тема: Записи. Записи с вариантами.
Пример. Объявить запись Student и объект TStudent, содержащие поля: ФИО студента, номер курса и номер группы. Продемонстрировать доступ к полям записи и объекта (доступ к полям объекта осуществить через методы).

```
program recobj;
type
(* запись "студент"
содержит три поля *)
Student = record
  sFIO: string;
  iCourse: byte;
  sGroup: string;
end;
(* объект "студент"
содержит три поля и два метода *)
TStudent = object
  fsFIO: string;
  fiCourse: byte;
  fsGroup: string;
  procedure SetStudent (sFIO: string; iCourse: byte; sGroup: string);
  procedure GetStudent (var sFIO: string; var iCourse: byte;
    var sGroup: string);
end;
var
  plushkin: Student; { переменная типа запись }
  kirpichov: TStudent; { экземпляр объекта }
  sFIO, sGroup: string;
  iCourse: byte; { переменные для записи значений полей }
(*-----------------------------------
Метод: задание свойств студента
-----------------------------------*)
procedure TStudent.SetStudent (sFIO: string; iCourse: byte; sGroup: string);
begin
  fsFIO := sFIO;
  fiCourse := iCourse;
  fsGroup := sGroup;
end;
(*-----------------------------------
```

```
Метод: получение свойств студента
-----------------------------------*)
procedure TStudent.GetStudent (var sFIO: string; var iCourse: byte;
  var sGroup: string);
begin
  sFIO := fsFIO;
  iCourse := fiCourse;
  sGroup := fsGroup;
end;
(*-----------------------------------
Сборка: проверка работы записи и объекта
-----------------------------------*)
begin
  { обращаемся к полям записи }
  writeln('These are record''s fields');
  plushkin.sFIO := 'Plushkin Ivan';
  plushkin.iCourse := 1;
  plushkin.sGroup := '20-103';
  writeln (plushkin.sFIO,': ',plushkin.iCourse,' year, ',plushkin.sGroup);
  { обращаемся к полям объекта - строго через методы}
  writeln('These are object''s fields and methods');
  kirpichov.SetStudent ('Kirpichov Alexey',2,'20-201');
  kirpichov.GetStudent(sFIO, iCourse, sGroup);
  writeln (sFIO,': ',iCourse,' year, ',sGroup);
  { плюха Паскаля: обращение к полям объекта напрямую }
  writeln (kirpichov.fsFIO,': ',kirpichov.fiCourse,' year, ',kirpichov.fsGroup);
  { завершаем программу }
  writeln('Press any key');
  readln;
end.
```

Задания:

1. Создать тип с описанием товара: код, наименование, цена за единицу товара, количество и сумма. Вывести наименование и цену товаров, коды которых лежат в диапазоне 100 … 120.
2. Создать тип с описанием товара: код, наименование, цена за единицу товара, количество и сумма. Подсчитать итоговые суммы по каждому товару.
3. Создать тип с описанием товара: код, наименование, цена за единицу товара, количество и сумма. Вывести название товара с наибольшей ценой.
4. Определить тип для представления анкеты школьника, включающей в себя ФИО, возраст, номера школы и класса и оценки по каким-то пяти предметам. Выдать сведения о школьнике с наивысшим средним баллом.
5. Ввести информацию о десяти студентах: фамилия, пол, год рождения, год поступления, оценки по шести предметам. Подсчитать число студентов мужского пола, родившихся в xx году.
6. Ввести информацию о десяти студентах: фамилия, пол, год рождения, год поступления, оценки по шести предметам. Вывести анкетные данные студентов-отличников.

7. Ввести информацию о десяти студентах: фамилия, пол, год рождения, год поступления, оценки по шести предметам. Вывести анкетные данные студентов, получивших оценки 2.
8. Ввести информацию о десяти студентах: фамилия, пол, год рождения, год поступления, оценки по шести предметам. Вычислить средний балл группы и вывести на экран список студентов, имеющих средний балл ниже среднего балла группы.
9. Ввести информацию о десяти студентах: фамилия, пол, год рождения, год поступления, оценки по шести предметам. Упорядочить список студентов по среднему баллу.
10. Ввести информацию о десяти студентах: фамилия, пол, год рождения, год поступления, оценки по шести предметам. Упорядочить список студентов по фамилии.
11. Описать тип с данными о сотрудниках (структуру записи продумайте самостоятельно). Вывести список сотрудников, проживающих по улице "Победа".
12. Описать тип с данными о сотрудниках (структуру записи продумайте самостоятельно). Подсчитать количество сотрудников, имеющих детей.
13. Описать тип с данными о сотрудниках (структуру записи продумайте самостоятельно). Подсчитать количество подарочных наборов, которые необходимо приобрести для детей сотрудников.
14. Описать тип с данными о сотрудниках (структуру записи продумайте самостоятельно). Начислить премию в размере 50% от оклада сотрудникам, стаж работы которых на данном предприятии превышает 10 лет.
15. Описать тип с данными о сотрудниках (структуру записи продумайте самостоятельно). Напечатать список сотрудников, которые в текущем году отмечают круглую дату.
16. Описать тип с данными о сотрудниках (структуру записи продумайте самостоятельно). Напечатать список сотрудников, у которых день рождения в августе.
17. Точки заданы координатами x и y. Выдать координаты точки, наиболее близко расположенной к началу координат.
18. Вершина характеризуется названием и высотой. Выдать название самой высокой вершины.
19. Круг задается радиусом и координатами центра. Определить, найдется ли среди данных десяти кругов круг, лежащий внутри данного круга.
20. Круг задается радиусом и координатами центра. Определить, найдется ли среди данных десяти кругов круг, лежащий вне данного круга.

Лабораторная работа №6
Тема: Файлы в Паскале
Пример. Программа быстрого копирования файлов

```
Uses Crt;
Var
FromF, ToF : File;
```

```
NumRead, NumWritten : Word;
Buf : Array [1..2048] Of Char;
Begin
{ Открываем входной файл }
Assign(FromF, ParamStr(1));
Reset(FromF, 1);
{ Размер буфера записи = 1 байт }
{ Открываем выходной файл }
Assign(ToF, ParamStr(2));
ReWrite(ToF, 1);
{ Размер буфера записи = 1 байт }
WriteLn('Копирую ', FileSize(FromF), ' байт...');
Repeat
BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
BlockWrite(ToF, Buf, NumRead, NumWritten);
Until (NumRead = 0) Or (NumWritten <> NumRead);
Close(FromF);
Close(ToF);
end.
```

Задания:

Дан файл содержащий массив из натуральных чисел. Составить программу, которая записывает в файл результат сортировки его элементов следующим образом:

1. Сначала все элементы, начинающиеся цифрой 1, затем – цифрой 2, затем – цифрой 3 и т.д.
2. Сначала все элементы, заканчивающиеся цифрой 1, затем – цифрой 2, затем – цифрой 3 и т.д.
3. Все элементы на нечётных местах – по возрастанию, остальные – по убыванию.
4. Сначала по возрастанию - все кратные 2, затем, все кратные 3, затем, все кратные 5, затем, все кратные 7, затем все остальные по убыванию.
5. Сначала все простые числа – по возрастанию, затем все остальные – по убыванию.
6. Сначала все нечётные – по возрастанию, затем все нечётные по возрастанию.
7. Половина массива по возрастанию, половина – по убыванию.
8. По убыванию, методом прямого выбора.
9. По убыванию, методом пузырька.
10. Сначала идут элементы, кратные числу K (дают 0 в остатке от деления на K), затем те, которые при делении на K дают в остатке 1, и т.д. Последнюю группу составляют числа, остаток от деления на K которых равен K-1.

Лабораторная работа №7, 8

Тема: Текстовые файлы. Работа с текстом.

Пример. Программа считающая количество пробелов в заданном файле.

```
var
T: Text;
```

```
C: Char;
Spaces: Word;
S: String[79]; { 79-макс. длина пути в DOS }
begin
Write('Введите имя файла: ');
Readln(S);
Assign(T, S);
{ открываем файл для чтения }
{$I-}
Reset(T);
{$I+}
{ если не нуль, то была ошибка }
if IOResult <> 0 then
begin
Write('Error when open file ', S, ' !');
Halt;
end;
{ иначе все в порядке, продолжаем }
{ ЦИКЛ: пока НЕ КОНЕЦ ФАЙЛА }
While (not Eof(T)) do
begin
{ читаем из файла переменную }
Read(T, C);
{ если пробел, увеличиваем счетчик }
If C = ' ' then Inc(Spaces);
Write(C);
end;
Writeln('КОЛИЧЕСТВО ПРОБЕЛОВ: ', Spaces);
Readln;
end.
```

Задания:

1. Написать программу, проверяющую, является ли слово из данного файла палиндромом (пишется одинаково в обе стороны) и записывающую результат в выходной файл в виде: “ палиндромом” либо “не палиндромом ”.
2. Написать процедуру, заменяющую звёздочками все слова, состоящие из трёх букв из исходного файла и записывающую результат в выходной файл.
3. В файле содержится две строки, изъять из первой строки все слова, которые встречаются во второй и записать результат в выходной файл.
4. Программа должна считывать набор символов из файла и затем запрашивать строку на ввод. Проверить, можно ли из введенных символов составить данную строку и записать результат в выходной файл в виде: “построение строки возможно: строка” либо “построение строки не возможно!”.
5. Считать из файла последовательность латинских букв. Напечатать эту последовательность, упорядоченную по алфавиту. Результат записать в выходной файл.

6. Найти в строке считанной из файла слова, начинающиеся и заканчивающиеся на одну и ту же букву. Результат записать в выходной файл.
7. Считать из файла символьную строку. Найти длину самого длинного и самого короткого слова, найденные слова и их длину записать в выходной файл.
8. Подсчитать количество слов в строке считанной из файла и записать результат в выходной файл.
9. Подсчитать в строке считанной из файла число букв A и B, если букв A больше, чем B, то удалить в строке все символы B полученную строку и результат подсчёта записать в выходной файл.
10. Считать текст из файла на русском языке. Записать в выходной файл в алфавитном порядке все глухие согласные буквы, которые входят в каждое нечётное слово и не входят хотя бы в одно чётное слово.

Лабораторная работа №9
Тема: Работа с графикой. Инициализация графического режима.
Пример. Следующая программа выводит на экран названия всех адаптеров и диапазоны возможных номеров режимов их работы.

```
Uses Graph;
var
D,L,H: Integer;
const
N: array [1..11] of String [8] =
('CGA ', 'MCGA ', 'EGA ',
'EGA64 ', 'EGAMono ', ЧВМ8514 ',
'HercMono', 'ATT400 ', 'VGA ',
'PC3270 ', 'Ошибка ');
begin
WriteLn('Адаптер Мин. Макс.');
for D := 1 to 11 do
begin
GetModeRange(D, L, H);
WriteLn(N[D], L:7, H:10)
end
end.
```

Задание:
1.Написать программу, рисующую точку перемещающуюся по диагонали из одного угла экрана в другой и меняющую свой цвет, причем пройдя этот путь точка должна возвращаться в исходный угол экрана и повторять движение.
2. Написать программу, рисующую точку перемещающуюся по горизонтали от одной стенки экрана к другой и изменяющую свой цвет, причем пройдя этот путь точка должна возвращаться в исходное положение на экране и повторять движение.

3. Написать программу, рисующую точку в центре экрана которая постепенно бы превращалась в горизонтальную линию и изменяла свой цвет, а затем опять становилась бы точкой

4.Написать программу, рисующую точку в центре экрана которая постепенно бы превращалась в окружность увеличивающуюся в размерах и уходящую за пределы экрана меняя свой цвет, а затем опять становящуюся точкой.

6. Написать программу, изменяющую цвет фона экрана в соответствии с порядком цветов в радуге.

7. Создать подобие кругов на воде используя пять окружностей с центром в одной точке и разными радиусами постоянно меняющими свой цвет на следующий в палитре в соответствии с возрастанием радиуса.

8. Написать программу, рисующую окружность за пределами экрана которая постепенно бы превращалась в точку и изменяла свой цвет, а затем опять повторяющую данное действие.

9. Вывести на экране произвольное изображение в различных граффических режимах, учитавая то, что некоторые режимы могут вызвать ошибки.

Лабораторная работа №10, 11

Тема: Построение графиков функций. Построение графических примитовов.

Пример 1. Написать программу для построения графика функции $y = \left|\frac{x}{x-2}\right| - 1$

```
Uses Graph,crt;
const
{ OPTIONS } {далее следует секция параметров графика: границы, цвет и т.д.}
shag=0.0001;
lgr = -50;
pgr = 50;
zcrtX=320;
zcrty=240;
mtrX=10;
mtrY=10;
colorG=15;
colorOXY=4;
OXYminX=-200;
OXYmaxX=200;
OXYminY=-200;
oxymaxY=200;
Var grDriver : Integer;
    grMode : Integer;
  x,y:real;{extended;}
     i:integer;
Begin
grDriver:=Detect;
InitGraph(grDriver, grMode, '');
for i:=OXYminX to OXYmaxX do putpixel(zcrtX+i,zcrty,colorOXY);
for i:=OXYminY to OXYmaxY do putpixel(zcrtX,zcrty-i,colorOXY);
x:=lgr;
```

```
while x<=pgr do
begin
{Function} {здесь задаём искомую функцию}
 if x<>2 then begin y:=abs(x/(x-2))-1;
 putpixel(zcrtx+trunc(x*mtrX),zcrty-trunc(y*mtrY),colorg) end;
 x:=x+shag;
end;
readkey;
closegraph;
end.
```

Задания: построить графики следующих функций:

1 $x^x e^{-x^2}$

2 $\frac{Tg(x)}{x} e^{-x}$

3 $\frac{Ln(1+\pi x)}{x} e^{-x}$

4 $\frac{1-e^x}{Sin(x)}$

5 $\frac{Sin(x)}{x} e^{-x}\left(1+x^3\right)$

6 $\frac{Cos(x)}{\pi/2-x}$

7 $\frac{x(1+x)^3}{(1+x)^3-1}$

8 $\frac{Ln(1+x)}{x} - \frac{Sin^2(x)}{x^2}$

9 $e^{1-x^2} Ln\left(1+x^2\right)$

10 $Tg(3ArcTg(x))$

11 $Sin(3ArcSin(x))$

12 $Cos(3ArcCos(x))$

13 $(1-x)^5 e^{xSin(x)}$

14 $\frac{(1-x)\left(1-x^2\right)}{1-x^5}$

15 $x^{6,5} e^{-Cos(x)}$

16 $x^3 e^{-Sin(\pi x)}$

17 $x^2 Ln(x) e^{-x}$

18 $\frac{ArcTg(x)}{x} e^{-x}$

19 $x^x e^{-x}$

20 $\frac{Sin(x)}{x} e^{-x}$

5 семестр

Лабораторная работа № 1.

Тема: Переменные.

Переменные могут быть числовыми, текстовыми и других типов. Название переменной начинается с латинской буквы, далее могут быть буквы и цифры (не более 31 символа). Строчные и прописные буквы различаются. Не допускается совпадения имени переменной с названием команд.

Числовые переменные. Это числа, векторы, матрицы и многомерные массивы. В компьютере все числа представлены примерно с 16 десятичными знаками, под каждое вещественное число отводится 8 байтов, под комплексное число – 16.

Ввод чисел.

Целые числа. В системе они не выделяются явно. Набрать и выполнить отдельно каждую команду:

```
a=2      a=2.0        a=2;
a=1:6        b=1:20       c=10:-2:5
```

Разобрать самостоятельно: Командное окно. Командная строка. Редактирование командной строки. Буфер исполненных команд. Как выбирать информацию из командного окна и из буфера исполненных командных строк.

Вещественные числа. Набрать и выполнить отдельно каждую команду:

```
d=0.5:0.3:2.5        d=.5:.3:2.5
d=.5+1:.3-.1:2.5*2       length(d)    d(end)
d(end-2)     d(1)    d(0)     d(2:7)   d(7:-1:2)
d(150)       f=linspace(1.5,30,143)   length(f)
```

Индексы всегда начинаются со значения 1. Команды набираются строчными буквами. Возможна многопараметричность команд.

Константы системы MATLAB (их не следует “портить”):

```
realmax  realmin      pi       i        j        eps
```

Комплексные числа. Набрать и выполнить отдельно каждую команду:

```
q=1+2*i      q=1+2i       real(q)      imag(q)
abs(q)   conj(q)      s=angle(q)  %здесь -π<s≤π
q=1+2*i; r=3;     fi=0:.01:pi;
z=q+r*exp(i*fi); plot(z)  %верхняя полуокружность
```

Ввод векторов.

Вектор-строка:

```
a=1:6 linspace(1,6,10)  %применима для задания только
```

вещественных векторов

Вектор-столбец:

```
a=(1:6)' linspace(1,6,10)'.
```

Ввод матриц.

$A(i, j)$ – элемент из i-й строки и j-го столбца. $A(k)$ – k-й элемент таблицы, вытянутой в столбец. Набрать и выполнить отдельно каждую команду:

```
A=[1, 2; 3; 4]        A=[1; 2; 3; 4]        A(2, 2)
A(3)     A(5)     size(A)      A(3,4)=10    size(A)
A(5)=6        size(A)      A(22)=3      A=A(:)
A(22)=3  size(A)       [m,n]=size(A)
A=reshape(1:24,4,6)   size(A)
A([1,end],:)=[]       A(:,[1,end])=[] size(A)
```

Некоторые специальные матрицы.

Набрать и выполнить отдельно каждую команду:

```
m=3; n=4; eye(m, n)       eye(m)        eye(n)
ones(m, n)        ones(m)       ones(n)       zeros(m, n)
rand(m, n)        rand(m, n)        rand('state', 0)
rand(m, n)        rand(m)  %равномерное распределение на (0, 1)
randn(m, n) randn('state', 100)  % нормальное распределение
v1=1:4        v2=7:12       toeplitz(v1,v2)
    toeplitz(v1)
```

Некоторые простые команды.

Набрать и выполнить отдельно каждую команду:

```
A=reshape(1:24,4,6)   triu(A)       triu(A,0)
triu(A,2)         triu(A,-1)        tril(A)
v=1:5         diag(v)       diag(v,2)         diag(v,-1)
diag(A)       diag(A,2)         diag(A,-1)
A=reshape(1:24,4,6)   rot90(A)      rot90(A,2)
```

Выдачи на экран.

Команда format. По умолчанию (format short) выдается 5 знаков, для целых чисел 9 знаков, порядки изменяются от -308 до +308, опция format long e – 16 знаков. Опция format short e позволяет получать ровные столбцы. Набрать и выполнить отдельно каждую команду:

```
a=2      a=.001       a=1e-3       a=1e-5
a=123456789      a=1234567891      a=1+3*i
format long e, 2^.5, format short
```

Текстовые переменные. Символ занимает 2 байта. Их берут в кавычки и используют для задания заголовков в числовых выдачах и на графиках, для задания формул и т. д. Можно переводить текстовые переменные в числовые и наоборот. Выполнить в командной строке:

```
t='График функции sin(x)'
```

Контроль за переменными. Набрать и выполнить отдельно каждую команду:

```
who          whos
```

Система помощи.

demo %запускает режим демонстрации для основных команд системы MATLAB

help %выдает список директорий системы

help имя директории %выдает список команд директории

help имя команды %выдает описание команды

type имя команды %выдает текст команды или программы пользователя, если он составлен в терминах MATLAB

save имя_файла %сохранить рабочую область в файле

load имя_файла %загрузить рабочую область из файла

clear %очистка рабочей области

exit %выход из MATLAB

Упражнения.

1. Из заданной матрицы A выбрать вектор, компоненты которого есть все окаймляющие A элементы, взятые в порядке $A(1,1)$, $A(2,1)$, … , $A(1,2)$.

2. Из заданной матрицы A размера $m\times n$ построить матрицу B с m строками, у которой диагонали с номерами 0, 1,…, n-1 были бы столбцами A с номерами 1:n, а все остальные элементы равнялись бы нулю.

3. Построить последовательность из 1000 целых случайных чисел, которые равновероятно принимали бы значения 11:18, и выдать, сколько из них оказалось равным 11, 12, …, 18. Используйте команду round(k) округления k до ближайшего целого.

Лабораторная работа № 2.

Тема: Элементы XY-графики.

Открытие графического окна.

```
figure          whitebg      zoom on
```

Пример. Построим график функции y=sin(2πx), 0≤x≤5:

```
x=0:1e-3:5; y=sin(2*pi*x); plot(y) plot(x, y), grid
```

Использование режима zoom:

```
k=100; y=sin(2*pi*k*x); plot(y)
```

Автоматическое чередование цвета. Теперь будем, как правило, нумеровать строки.

```
x=linspace(0,1,20); k=.1:.1:.8; y=k'*x; plot(x,y)
```

Здесь определяется вектор-строка x=0:20, затем вектор-строка k из 8 угловых коэффициентов, далее получается матрица $y=k'\cdot x$ как произведение вектора-столбца k' на вектор-строку x. Строки этой матрицы состоят из точек соответствующих прямолинейных отрезков. Наконец, строятся графики этих отрезков как функций от x. Видно, что цвета, которых всего 7, чередуются в следующем порядке: желтый (yellow), фиолетовый (magenta), голубой (cyan), красный (red), зеленый (green), синий (blue), белый (white).

Рассмотрите третий текстовый аргумент команды plot:

```
plot (x, y, 'g.')
```

Все кривые на рисунке станут зелеными, а линии будут изображаться отдельными точками. Аналогично можно задать любой другой цвет и задать кривую в виде пунктира.

Полярные координаты. Набрать и выполнить команды:

```
x=1:.01:3;          nx=length(x);       r=x.^2;
fi=linspace(0,5*pi,nx); polar(fi,r)
```

Пример. Построение многозначной функции:

```
x=0:.1:6*pi; y=cos(x); plot(x,y) plot(y,x)
```

Команды управления осями координат. Рассмотреть опции команды axis:

```
axis off axis on axis ([-10,10,-5,20]) axis auto axis
equal axis square
```

Размеры осей можно задавать и для трехмерной графики, но цвет в ней используется для характеристики величины ординаты и команда zoom не работает.

Лабораторная работа № 3.

Тема: Примеры использования системы MATLAB.

Суммирование. Пример 1: найти для заданного n частичную сумму ряда $S_n = 1/k^2$, k=1..n:

```
n=100 (1000);          k=1:n;        f=k.^(-2);
plot(cumsum(f))        [sum(f),pi^2/6]
```

Команда cumsum(*f*) подсчитывает все частичные суммы S_k (*f*) для всех k=1..n, так что на графике можно наблюдать процесс формирования нужной нам величины. Далее вычисляется численный и точный результат. При n=1000, получим S_n = 1.6439, т.е. ошибка .001.

Сходимость не всегда столь очевидна, как на этом графике. Например, для заданного n и m>1 найти частичную сумму ряда $S_{(n,m)} = \sum 1/k^m$, k=1..n (при m=1– *гармонический* ряд).

```
m=2 (1.5/1.2);          n=1000 (1e4);          k=1:n;
f=k.^(-m);          plot(cumsum(f))        sum(f)
```

При m=1.5, глядя на график, нет полной уверенности в сходимости ряда, и тем более для m=1.2: $S_{(1000,m)}$ = 4.3358, $S_{(1e4,m)}$ =4.7991. Факт сходимости ряда при m=1.01 нельзя установить численно из-за низкой скорости его сходимости.

Пример 2. Найти $I = \int x \cdot dx/\sin(x)$, $x \in [0, 3]$. Ясно, что функция $x/\sin(x)$ непрерывная.

```
n=100;          h=3/n;          x=h/2:h:3-h/2;
f=x./sin(x); plot(h*cumsum(f)), grid, sum(h*f)
```

т. е. аппроксимировать *f* в серединах интервалов (точки *x* называют *полуцелыми* в отличие от концов счетных интервалов – *целых* точек). Сравнив ответ 8.4495 с графиком можно решить, что сходимость еще не достигнута, но при n=1e3 I = 8.4552, так что при n=100 со сходимостью в действительности все в порядке, а возрастание функции h*cumsum(*f*) на правом конце происходит из-за роста там функции *f* – это видно по ее графику **plot(f)**.

Для матрицы A команды sum и cumsum работают вдоль столбцов (значит, по первому индексу), а для вектора – вдоль него независимо от того, строка это или столбец. Чтобы провести суммирование для матрицы A вдоль ее строк, нужно выполнить sum(A,2), т.е. указать для выполнения команды второй индекс. Это правило относится ко многим командам MATLAB и к многомерным матрицам тоже – по умолчанию имеется в виду первый индекс, а в противном случае нужно всегда указывать, по какому индексу должна работать команда, и это указание не сохраняется для последующих команд.

Произведения. С помощью команд prod и cumprod вычисляются и обрабатываются произведения. Например, найдем $\prod(1-1/k^2)$, k=2..100 (при $k \to \infty$ $\prod \to 1/2$):

```
n=100;   k2=(2:n).^2;   a=1-1./k2;   cp=cumprod(a);
cp(end), plot(cp/.5), grid
```

Результат cp(end)=0.5050 говорит о том, что сходимость здесь не очень быстрая. Это видно и из графика, на котором представлена относительная ошибка результата.

При вычислении произведений можно выйти за числовую шкалу. Найдем, например, для каких k можно вычислить $k!$. Ясно, что $k \le 200$, тогда имеем

```
n=200; k=1:n; kf=cumprod(k); plot(kf)
```

Из-за быстрого возрастания *kf* и ограниченной разрешимости дисплея (не более 0.5% от максимального значения на графике) мы видим всего одну точку, перед которой, как нам ошибочно кажется, идут нули и за которой идут числа inf (infinity), вообще никак не представленные на рисунке. Точно так же графика обходится и с переменной NaN (not a number), и это обстоятельство может быть иногда полезным. Переменная NaN возникает в случаи неопределенности: *0/0 inf-inf inf/inf*. Переменные inf и NaN (со знаком ±) можно использовать в программах. Для нахождения максимального k выполним команду: **sum(isinf(kf))**, в которой isinf(*kf*) выдаст 1 на тех позициях вектора размеров *kf*, где элементы *kf* есть inf, и 0 на остальных позициях. Так как ответ =30, то $k_{max}=n-30=170$, что можно было бы получить и сразу, выполнив:

```
km=sum(isfinite(kf))
```

Здесь isfinite отмечает те элементы числовой переменной, которые отличны от inf и NaN. При выходе произведения за числовую шкалу для сомножителей можно использовать функции: log (натуральный логарифм), log10 (десятичный логарифм), abs (модуль), sign (знак).

Логические задачи. Пример 1. Найти общие элементы двух векторов:

```
x=1:20; y=15:30; [X,Y]=meshgrid(x,y); v=X(X==Y)
```

Пример 2. Взяв на сторонах единичного квадрата по 200 интервалов, найти, сколько точек получившейся таким образом сетки попадает внутрь вписанной в него окружности.

```
tic,    x=0:1/200:1;              [X,Y]=meshgrid(x);
M=abs(X+i*Y-.5-i*.5)<1/2;    s=sum(M(:)), t1=toc
```

Ответ *s*=31397 точек, t1=0.61 сек. Использование цикла for менее целесообразно:

```
tic,  s=0;  w=1:201;  for  I=w,  for  J=w,  if
norm([x(I),x(J)]-.5)<.5,  s=s+1;  end,  end,  end,  s,
t2=toc
```

Результат t2=2.69 сек., что в t2/t1=4.4 раза дольше.

Упражнения

1. Усовершенствуйте последний пример и посмотрите, как изменится значение t2/t1.

2. Проверьте, что при вычислении определителя квадратной матрицы A порядка n методом Гаусса нужно выполнить $\cong 2n^3/3$ операций сложения и умножения. Считая их одинаковыми по времени выполнения, найдите, за какое время t выполняется 1 млн. операций, взяв n=500 и выдав время счета det(A).

3. Используя предыдущий результат, оцените число арифметических операций для вычисления функций sin и exp.

4. Оцените число операций, которое затрачивается в MATLAB для организации одного шага в цикле for-end.

5. Используя операторы max, diff и sort, составить программу, которая для заданной квадратной матрицы A определяла бы, что A является нижней треугольной с ненулевыми элементами на главной диагонали или получается из таковой перестановкой столбцов.

Лабораторная работа № 4.

Тема: Графический способ решения уравнений.

Пример 1. Найти корни уравнения $x \cdot \sin(x^2)=0$ на отрезке [0,3]. Выполнить команды:

```
x=0:.01:3;   f=x.*sin(x.^2);   plot(x,[f;0*f]),   grid
ginput
```

В команде ginput точка снимается нажатием левой клавиши мыши, Enter – выход из ginput. Другой способ:

```
nx=length(x);     w=1:nx-1;     x(find(f(w).*f(w+1)<0|
f(w)==0))
```

Или упростив:

```
nx=length(x); w=1:nx-1; x(f(w).*f(w+1)<0|f(w)==0)
```

Пример 2 (неявные функции). Построить график функции $f(x, y)\equiv x^3y-2xy^2+y-0.2=0$, $x, y\in[0, 1]$.

```
h=.02;    x=0:h:1;    [X,Y]=meshgrid(x);    f=X.^3.*Y-
2*X.*Y.^2+Y-.2; v=[0,0]; contour(x, x, f, v), grid
```

На графике зеленая линия (она двузначная) представляет искомый результат. Область в первом квадранте между этими кривыми обозначим через G. Выясним, какой знак имеет f в области G:

```
mesh(x,x,f.*(f>0))
```

И это пример трехмерной, т.е. xyz-графики. Найдем площадь S этой области: `S=h^2*sum(f(:)>=0)`.

Найдите S, при h=0.01, h=0.005. При интегрировании всегда естественно делать такие проверки.

Определим, какой объем заключен между поверхностью $f(x, y)$ и областью G, где $f(x, y)\geq 0$:

```
V=h^2*sum(f(f>=0))
```

Найдите S, при h=0.01, h=0.005. Теперь не нужно писать f(:), поскольку f ($f\geq 0$) – вектор.

Команда отправляет числовую информацию о графике в матрицу C и строит график, выбрав значения уровней автоматически, а из матрицы C можно последовательно выбирать все кривые:

```
C=contour(x,x,f); clabel(C)
```

Обобщения. Графическим способом можно решать системы уравнений и уравнения в комплексной плоскости. Команда contour3 строит линии уровней для функций $f(x,y,z)$, при этом сетки по аргументам всегда должны быть прямоугольными.

Упражнения.

1. Построить с помощью команды contojur график эллипса $x^2/a^2+y^2/b^2=1$, найти его площадь и сравнить ее с точным значением $S=\pi a \cdot b$.

2. На следующем примере рассмотреть команду subplot:

```
x=0:.01:1;                          for k=1:4,
subplot(2,2,k),  plot(x,sin(pi*k*x)),                end
subplot(1,1,1)
```

Лабораторная работа № 5.

Тема: Полиномы.

По степени применимости, по разнообразию и качеству соответствующих команд скалярные полиномы – следующие за матрицами математические объекты в MATLAB. Полином $p(x)=a_n x^n + a_{n-1} x^{n-1} +\ldots+a_0$ задается вектор–строкой p из коэффициентов a_n, a_{n-1},…, a_0. Его степень задавать не надо, так как n=length(p)–1 (полином есть константа, при n=0); коэффициенты – любые комплексные числа. Вектор p интерпретируется системой как полином только тогда, когда он задается в качестве параметра для одной из команд, производящих вычисления с полиномами. Так как в этих командах не проверяется условие $a_n\neq0$, надо стараться самим соблюдать его, поскольку иногда это может служить источником ошибок.

Основные команды для действий с полиномами следующие:

conv(p,q) – произведение полиномов. Название команды convolution (свертка), ибо коэффициенты произведения получаются как компоненты свертки векторов p и q.

[q,r]=deconv(b,a) – частное (q) и остаток (r) от деления b на a, так что conv(a,q)+r=b.

residue(b,a) – разложение рациональной функции $b(x)/a(x)$ на элементарные дроби над полем комплексных чисел с выделением целой части. Если $a(x)$ имеет кратные или близкие друг к другу корни, результаты могут быть неверными, так как такая задача плохо обусловлена, т. е. есть сильная зависимость результата от коэффициентов.

p=poly(r) – построение полинома по корням, заданным в векторе-столбце r. Для квадратной матрицы r полином p будет ее характеристическим многочленом.

polyval(p,x) – поэлементное вычисление значений полинома p на множестве x, где x – вектор, или матрица. Размеры результата совпадают с size(x).

polyder(p) – производная от p.

roots(p) – корни полинома (их порядок не определен; как и для функции residue, возможна неустойчивость результата). Корни полинома вычисляются как собственные значения некоторой матрицы $A(p)$ того же порядка.

Пример 1. Построить график уравнения $p(x)=x^3–x+2$ на отрезке $-1\leq x\leq1$:

```
p=[1,0,-1,2]; x=-1:.01:1; f=polyval(p,x); plot(x,f), grid
```

Уравнение не имеет корней на заданном отрезке. Найдя корни: **`roots(p)'`**, убеждаемся в этом.

Пример 2. Разделить предыдущий полином на x–3:

```
[q,r]=deconv(p,[1,-3])
```

Ответ q = 1 3 8, r = 0 0 0 26 означает, что частное $q(x)=x^2+3x+8$, а остаток r=26.

Пример 3. Разложить функцию $(x–3)/p(x)$ на элементарные дроби:

```
[r,s,k]=residue([1,-3],p); r', s', k'
```

В ответе векторы r', s' состоят из трех элементов, а k=[], т. е. целой части в разложении нет; действительно, у числителя первая, а у знаменателя третья степени. Элементы векторов r и s означают, что $(x\text{-}3)/p(x)= \sum r_i/(x\text{-}s_i)$, i=1..3. Функция residue работает и в обратную сторону; восстановим исходный числитель и знаменатель:

```
[q,p]=residue(r,s,k)
```

Пример 4 (Уилкинсон, 1963). Иногда, несмотря на хороший “разброс” корней полинома, их вычисленные значения могут очень сильно зависеть от значений некоторых коэффициентов потому, что производные корней по этим коэффициентам – очень большие по модулю числа. Такие задачи называются плохо обусловленными и всегда требуют повышенного внимания. Пусть v_n=1..n, где n=2, 3, …, а p_n=poly(v_n') – полином с корнями 1..n, которые хорошо отделены друг от друга, а w_n=roots(p_n) – вектор-столбец с вычисленными корнями полинома p_n. Сравним v_n' и w_n для различных n. При n=2:

```
n=2; vn=1:n; pn=poly(vn'); wn=roots(pn); [vn',wn]
```

Результат будет [1 2; 2 1] откуда видно, что элементы в w_n нужно упорядочить. Для n=3 имеем:

```
n=3; vn=1:n; pn=poly(vn'); wn=roots(pn); R=[vn',
sort(wn)]
```

Появившиеся в первом столбце R нули как бы “наведены” значениями из второго столбца (это следствие команды format), нули во втором столбце R говорят о влиянии погрешности в нахождении корней. Найдём её:

```
((R(:,2)-R(:,1))./R(:,1))'
```

Относительная погрешность =1e-15*(0.8882 -0.9992 0.2961), т. е. верны примерно 14 знаков. Для n=10 имеем:

```
n=10; vn=1:n; pn=poly(vn'); wn=roots(pn); R=[vn',
sort(wn)]; R1=(R(:,2)-R(:,1))./R(:,1)
```

Видим, что точность результата постепенно падает. Функция **me=max(abs(R1))** даст максимальную погрешность 6.3604e-10, т. е. теперь для всех корней верны только 9 знаков. Но корни еще остаются вещественными, поскольку; действительно, функция **iwn=sum(abs(imag(wn)))** дает 0. Далее, для n=20 получим me=0.0049, т. е. для всех корней верны только 2 знака, но результат еще вещественный. Сравним точные и вычисленные корни на графике: **plot(R), grid**, на котором видно, что погрешность для некоторых корней (11–18) уже видна на глаз.

При n=20 $a_{19}=pn(2)$=–210. Прибавим к нему 1e-7, т. е. внесем в него малое возмущение примерно в 10-м знаке, и повторим расчет:

```
n=20;   vn=1:n;   pn=poly(vn');   pn(2)=pn(2)+1e-7;
wn=roots(pn); R=[vn',wn], plot(R), grid
```

Сразу видим, что некоторые корни стали комплексными (мнимые части достигают по модулю 2.7), а iwn= 18.6684, и из графика. Если на графики убрать соединение между точками результат будет выглядеть более рельефно: **plot(R,'.'), grid**.

Выясним, почему так сильно изменились результаты при внесении столь малого возмущения. Обозначим через $p(x)$ наш *невозмущенный* полином p_n при n=20 и $a = a_{19}$: $p(x)$=prod(x-k), k=1..20, или $p(x)=x^{20}+ax^{19}+\ldots$ +20!, a=–210. Тогда для корней x=1..20 по теореме о производной неявной функции имеем $\partial p/\partial x \cdot ®x/®a + \partial p/\partial a = 0$, откуда $®x/®a = -\partial p/\partial a \,/\, \partial p/\partial x$. Так как $\partial p/\partial a = x^{19}$, а полином $\partial p/\partial x$ находится как polyder(pn). Поэтому для вычисления $®x/®a$ на множестве корней v_n надо выполнить команды:

```
n=20; vn=1:n; pn=poly(vn'); dpn=polyder(pn); dxda=-
(vn.^19)./                              polyval(dpn,vn);
plot(log10(abs(dxda)),'.'), grid
```

Так при x=8 $®x/®a=10^5$ и будет еще больше с ростом x. Поэтому внесение в коэффициент a возмущения должно обязательно сказаться на значениях некоторых корней. Более того, если эти необходимые изменения никак не проявились, метод следует забраковать.

Упражнение.

1. Для уравнения $p(x)=x^3 - 3.55x^2 + 5.1x - 3.1$ и найти графически его единственный вещественный корень. Сколько знаков вам удастся получить, уменьшая интервал около корня?

Лабораторная работа № 6.

Тема: Итерации.

Простые итерации в общем случае представляются в виде: $\boldsymbol{x}_{k+1}=\boldsymbol{F}(\boldsymbol{x}_k)$, k=0, 1, 2, …, $\boldsymbol{x}_0$ – начальное приближение, $\boldsymbol{F}$ – заданное отображение. Предел итераций, если он существует, обозначим через $\boldsymbol{x}$, и уравнение можно записать в векторной форме: $\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{X})$. Решения уравнения называются *неподвижными точками* отображения $\boldsymbol{F}(\boldsymbol{x})$. Все такие $\boldsymbol{x}_0$, для которых последовательность $\boldsymbol{x}_k$ сходится, образуют область сходимости итерационного процесса $\boldsymbol{F}$. Скорость сходимости итераций $\boldsymbol{x}_k$ характеризуется поведением последовательности $\boldsymbol{v}_k= \|\boldsymbol{x}_{k+1}–\boldsymbol{x}_k\| / \|\boldsymbol{x}_k–\boldsymbol{x}_{k-1}\|$. Для сходимости $\boldsymbol{x}_k$ уже не обязательно, чтобы существовал предел $\boldsymbol{v}$ последовательности $\boldsymbol{v}_k$, но очень часто для сходящихся итераций он существует, и если это так, то пусть $\boldsymbol{v}_+$:=abs($\boldsymbol{v}$). Тогда при $\boldsymbol{v}_+$=1 сходимость $\boldsymbol{x}_k$ будет крайне медленной; при 0<$\boldsymbol{v}_+$<1 – сходимость с геометрической прогрессией знаменателем $\boldsymbol{v}$, а при $\boldsymbol{v}_+$=0 сходимость быстрее геометрической прогрессии. Рассмотрим нелинейное уравнение: $(x–2)(x–3)=0$ или $f(x)\equiv x^2–5x+6=0$, корни $x_1=2$, $x_2=3$, построим для его решения несколько итерационных преобразований (схем) F и рассмотрим их работу.

Пусть для уравнения (2) $x=F(x)$, где $F(x)=(x^2+6)/5$. В этом случае появилось еще одно решение $x=\infty$. Потеря или приобретение новых решений случается при переходе от исходного уравнения к его итерационной форме: $x_{k+1}=F(x_k)$, k=1..n, где начальное приближение x_1 и число итераций n – заданы. Переменная $\boldsymbol{x}$ содержит значения x_k, x_t – текущее значение x_k. Тогда имеем:

```
xt=0; n=100; x=xt; TF='(xt^2+6)/5'; for k=1:n,
xt=eval(TF); x=[x,xt]; end, plot(x), grid
```

Функция eval она интерпретирует текстовую переменную TF как выражение $(x_t^2+6)/5$ и выполняет его. На графике нарисовано 101 значение x_k, итерации сходятся к x=2. Можно записать это и следующим образом:

```
xt=0; n=100; x=xt; TF='xt=(xt^2+6)/5;'; for k=1:n,
eval(TF), x=[x,xt]; end, plot(x), grid
```

Хотя внешне сходимость x_k к x_1=2 не вызывает сомнений, это в действительности всегда нужно проверять более тщательно. Так как $F'(x)=2x/5$, то $F'(2)=0.8<1$, а $F'(3)=1.2>1$, но итерации сходятся к такой неподвижной точке, для которой $|F'(x)|\le 1$. Отсюда ясно, почему x=2. Впрочем, не всегда можно найти $F'(x)$ аналитически, и поэтому в общем случае приходится использовать численный подход. При известных уже x_k имеем:

```
w=2:n; v=(x(w+1)-x(w))./(x(w)-x(w-1)); plot(v), grid
```

На графике видно, что v_k сходится к a=0.8, как и должно быть. Далее, изменяя начальное приближение повторите выполненное для следующих значений:

x_t=1.5, 1.9, 1.99, 1.9999 (в последних двух случаях на графиках появилась осцилляции справа, так как здесь разности $x_{k+1}-x_k$ и тем самым значения v_k теряют точность.

x_t=2 – график пуст, ибо все v_k=0/0=NaN.

x_t=2.01, 2.5, 2.9, 2.99, 2.9999 (в последнем случае x_k довольно долго задерживаются в районе неподвижной точки x_2=3 – они все время уходят от нее, но на графике этого не видно, а затем примерно за 60 итераций монотонно движутся к x_1=2).

x_t=3 – все x_k=3, график – пуст, так как $F(3)=15/5=3$ – никакой ошибки.

x_t=3.01 – предел для x_k и v_k будет ∞, т. е. x_2=3 является *неустойчивой* неподвижной точкой F, при малейшем сдвиге в пределе итераций получится, либо устойчивая неподвижная точка x_1, либо ∞ – приобретенная неподвижная точка. Давайте исследуем этот сдвиг.

x_t=3+1e-15, 3+1e-14, 3+1e-7: в первом случае ухода x_t еще нет, во втором – тенденция ухода уже зародилась и неизбежно произойдет с увеличением числа итераций, в третьем – проявился в полной мере уже на 100 итерациях.

x_t=-2.5, -2.9, -2.99, -3, -3.01, -100, 100. При x_t=–3 снова получим x_2 за одну итерацию, а при x_t=–3.01 получим ∞; таким образом, при вещественном x_0 итерации сходятся к устойчивой точке x_1=2 при $-3<x_0<3$ и к ∞, при $|x_0|>3$. Рассмотрим случай для $|x_0|$=3. Это реализовывают команды:

```
n=100;        fi=-pi:pi/20:pi;        xt=3*exp(i*fi);
TF='(xt.^2+6)/5';  for  k=1:n,  xt=eval(TF);  end,
plot(xt,'.')
```

На графике (комплекснозначном) отмечены 4 точки: z=2, z=3, точки $z=3\pm s\cdot$™, где $s\to 0$. Проверим результат при n=1000, отобразим результат: **`imag(xt')`**. Образы первой и последней точки начальной окружности слегка отличаются от z=2, а ее 21-я точка (она соответствует φ=0) есть z=3. Это происходит потому, что sin(π) и sin(-π) ≠ 0 в точности. Проверим результат с радиусом 3.01 (n=100), а затем команды: **`sum(isnan(xt))`** **`find(isnan(xt))`**. Получим, что точки первоначальной x_t с номерами 1, 21, 41 обращаются в ∞ (здесь NaN=∞/∞). Для радиуса 5 их будет 21, для радиуса 5.6 их будет 41, т.е. все.

Границу области сходимости обычно трудно исследовать аналитически, даже в таком простом примере, как этот, и сама она не определяется из условия $|F'(x)|=1$ или $|x|$=2.5. Условие $|\boldsymbol{F}'(\boldsymbol{x})|<1$ гарантирует устойчивость неподвижной точки $\boldsymbol{x}$ преобразования $\boldsymbol{F}$, условие $|\boldsymbol{F}'(\boldsymbol{x})|>1$ является достаточным для ее неустойчивости, в случае $|\boldsymbol{F}'(\boldsymbol{x})|=1$ необходимы дополнительное исследование. Неустойчивые неподвижные точки сравнительно редки в вычислительных задачах, но их полезно иметь в виду при исследовании численных алгоритмов. В нашем примере мы установили, что: множество $|x|\le 3$ принадлежит области сходимости итераций F, но не

описали её границу; при $|x|\geq 5.6$ итерации сходятся к ∞, и поэтому ∞ является устойчивой неподвижной точкой F. Исследуем границу области сходимости:

```
r=3:.1:5.6; z=[]; n=100; for kr=r, xt=kr*exp(i*fi);
for k=1:n, xt=eval(TF); end, z=[z;xt]; end
zn=isnan(z); Z=r'*exp(i*fi); plot(Z(zn)), axis equal
```

На графике границы видим на самом деле какая это “окружность”. Команда axis equal выбирает по осям одинаковый масштаб (действует только на текущий график). Построим границу четче задав шаг для φ равным pi/180, а для r=0.02, а затем выполнив команды, видим отличие области сходимости от круга $|z|\leq 3$:

```
zn=isnan([z;z(end,:)]); zn=diff(zn)~=0; plot(Z(zn)),
hold on
y=3*exp(i*fi); plot(y,'m'), axis equal, hold off
```

Теперь найдем те точки, которые перейдут в z=3 на первых 10 итерациях. Рассмотрим обратное отображение $x=\pm(5y-6)^5$, выполним для него 10 итераций, задав начальное приближение y=3:

```
n=10; yt=3; for k=1:n,yt=(5*yt-6).^.5; yt=[yt,-
yt];end, plot(yt,'.')
```

На графике получается практически та же линия. Более того, это верно и для других точек z из области сходимости F, но с некоторыми вкраплениями внутрь области сходимости. Задавать n еще большим нельзя, так как длина вектора y_t равна 2^n.

Пример 2. Представим уравнение $f(x)\equiv x^2-5x+6=0$ в виде: $x=F(x)$, где $F(x)=5-6/x$, т. е. $F'(x)=6/x^2$, $F'(3)=2/3<1$, $|F'(x)|<1$ при $|x|>2.45$. Следовательно, устойчивая точка – x_0=3. Найдем результат для “всех” $x_0=x_t$ ($\in \Re$):

```
n=100; xt=-5:.5:5; x=xt; TF='5-6./xt'; for k=1:n,
xt=eval(TF); end, plot(x,xt,'.')
```

Все пределы равны 3, выпадает только неустойчивая точка x_t=2, для которой итерации идут без ошибок округления. Возникает предположение, что вся комплексная плоскость с выколотой точкой x_1=2 стягивается итерациями в точку x_2=3, хотя не везде $|F'(x)|<1$. Посмотрим, как это выглядит на графике:

```
n=4; fi=-pi:pi/20:pi; xt=4*exp(i*fi); TF='5-6./xt';
z=xt; for k=1:n, xt=eval(TF); z=[z;xt]; end,
plot(z','.'), axis equal
```

Начальное приближение (окружность радиуса 4 с центром в нуле) итерируется 4 раза и все результаты выдаются на график. Окружности (на графике их 5) довольно быстро стягиваются в точку x_2=3. Используя zoom, посмотрите малые окружности. Далее, сделаем разрезы на начальной окружности:

```
n=4;        fi=-pi:pi/20:pi*.75;        fi(end-4)=[];
xt=4*exp(i*fi);   TF='5-6./xt';   z=xt;   for   k=1:n,
xt=eval(TF); z=[z;xt]; end, plot(z','.'), axis equal
```

Первая итерация меняет направление обхода окружности на противоположное, остальные – нет. Проверьте результат при n=20. Самая малая окружность лежит правее точки z=3 примерно в полосе от 3.0001 до 3.0004. Потерь и других новых неподвижных точек здесь нет.

Пример 3. Решим задачу методом Ньютона. Если x'' удовлетворяет уравнению $f(x'')=0$, а x' находится вблизи x'' и используется для приближенной аппроксимации $f(x'')$, то $0=f(x'')=f(x')+f'(x')(x''-x')$ или $x''=x'-f(x')/f'(x')$, при условии, что $f'(x'')\neq 0$ и тем самым $f'(x')\neq 0$. Это и есть итерации по Ньютону для уравнения $f(x)=0$. При переходе к индексной форме записи для $f(x)=x^2-5x+6$ и $f'(x)=2x-5$ получим $x_{k+1}=x_k-f(x_k)/f'(x_k)$ или $y=F(x)$, где $F(x)=x-f(x)/f'(x)$, $F'(x)=2f(x)/(f'(x))^2$. Поскольку $f'(x_{1,2})\neq 0$, то можем использовать метод касательных; а так как $F'(x_{1,2})=0$, можно ожидать быстрой сходимости, а решение $x_{1,2}$ будет устойчивым. Неподвижной точкой будет и inf, но она “чужая”, и природа ее гораздо сложнее. Набрать и выполнить команды:

```
TF='xt-(xt.^2-5*xt+6)./(2*xt-5)';
xt=0;    n=100;    x=xt;    for    k=1:n,xt=eval(TF);
x=[x,xt];end, plot(x), grid
w=2:n; v=(x(w+1)-x(w))./(x(w)-x(w-1)); plot(v), grid
```

Предел итераций при x_0=0 равен 2, а сходимость имеет порядок выше первого, ибо v_k, до потери ими точности, успели подойти к нулю. Оценим порядок сходимости (в смысле квадратичной сходимости):

```
w=2:n;v=(x(w+1)-x(w))./(x(w)-x(w-1)).^2;
plot(v(1:6)), grid
```

Теперь до потери точности v_k успевают подойти к 1 (у v_7 точность уже потеряна). Напомним, что точность теряется у v_k, но не у x_k. Чтобы увидеть потерю точности у v_k, задайте графике 7 точек. При x_0=4 предел итераций равен 3, а v_k успевают подойти k–1 (у v_6 точность уже потеряна), т. е. в зависимости от начального приближения $x_0=x_t$ итерации могут сходиться к обоим решениям задачи. Так как преобразование $F(x)$ уже не является дробно-линейным, рассмотрим трансформацию комплексной прямоугольной области в процессе итераций. Зададим область x_t из 41^2=1681 комплексных точек и проведем ее итерации (в командном окне будет сообщения о делении на нуль):

```
x=-10:.5:10; [X,Y]=meshgrid(x); xt=X+i*Y; n=100; for
k=1:n,xt=eval(TF);end, plot(xt,'.')
```

На графике видны оба решения задачи и некоторое множество точек на прямой Re(z)=2.5. Выполним 10 итераций и покажем графика для каждой итерации:

```
x=-10:.5:10; [X,Y]=meshgrid(x); xt=X+i*Y; n=10; for
k=1:n, xt=eval(TF); plot(xt,'.'), pause(1), end
```

Результат тот же, что и при 100 итерациях, но видим динамику его формирования. Полуплоскость Re(z)<2.5 стягивается точку x_1=2, а полуплоскость Re(z)>2.5 – в точку x_2=3: прямая Re(z)=2.5 переходит сама в себя. Точка x_1=2 имеет красный цвет потому, что в нее перешли первые 25 столбцов матрицы x_t, а остаток rem(25,7)=4, а 4-й цвет – красный, затем идут зеленые точки (26-й столбец) и синяя x_2=3, так как rem(41,7)=6, а 6-й цвет – синий. Проверить наши выводы численно:

```
z=xt(:);      [sum(abs(z-2)<.1),      sum(abs(z-3)<.1),
sum(abs(real(z)-2.5)<.1)]
```

Ответ даст 1025=41*25, 615=41*15, 38<41 на 3, а так как потерь в матрице x_t быть не может, выясним, во что перешли эти 3 точки - в inf или NaN:

```
z=xt(:,26); [sum(isinf(z)), sum(isnan(z))]
```

Найдем индексы этих трех точек (z_1=2.5-0.5i, z_2=2.5, z_3=2.5+0.5i) из 26-го столбца: `find(isnan(z))'`.

Теперь рассмотрим, как преобразуется прямая Re(z)=2.5; так как она переходит в себя, ее можно параметризовать задав y=Im(z), пусть y_k=Im(F^k(Re(z)=2.5), k=1, 2, …, т. е. после k итераций y переходит в y_k(y). Ясно, что на k-ой итерации в inf перейдут только те точки из всего множества y_{k-1}, которые равны нулю. Поскольку -inf<y<inf, на первой итерации это случится только с одной точкой z_1=2.5 (для нее y=0). Построив на графике y_1 после 1-й итерации, видим, что функция y_1(y) пересекает ось абсцисс только два раза (при этом y_1 дважды непрерывно пробегает от -inf до inf), так что на второй итерации в inf перейдут только две точки (эти точки z_1=2.5–0.5i, и z_3=2.5+0.5i):

```
xt=2.5+(-10:.1:10)'*i; y=imag(xt); n=1; for k=1:n,
xt=eval(TF); end, plot(y,imag(xt)), grid
```

Проверим результат при n=2 и снимем с графика четыре точки пересечения кривой y_2(y) с осью абсцисс:

```
q=ginput;q(:,1)'
```

Приблизительно они есть: -1.2075 -0.2068 .2068 1.2075, тогда как для y_1(y) это были значения –.5 и .5. Каждая переходящая в inf на k-ой итерации точка порождает слева и справа от себя две такие точки, которые перейдут в inf на (k+1)-ой итерации. Поэтому с ростом k точки y_k(y)=0, с одной стороны, уходят в обе стороны от нуля, а с другой – все чаще появляются в тех местах, где они однажды уже появились. Всего на k-ой итерации в inf перейдет 2^{k-1} точек. В действительности, с ростом k они всюду плотно заполняют прямую Re(z)=2.5, а между ними y_k(y) непрерывно пробегает все значения от -inf до inf. Убедимся в этом:

```
hy=1.0033e-4;  xt=2.5+(-1:hy:1)'*i;  w=2:length(xt);
n=100; for k=1:n, xt=eval(TF); end pzn=sum(sign(xt(w-
1)).*sign(xt(w))<0)
```

Ответ зафиксирует 674 перемены знака у функции $y_{100}(y)$ для $-1\le y\le 1$ с h_y=1.0033e-4 (из-за недостаточной малости шага не все они учтены). Для $-10\le y\le -8$ получим 677, для симметричного отрезка $8\le y\le 10$ получим 669. Ошибки при последовательном вычислении $F^k(z)$ на прямой Re(z)=2.5 будут ничтожными, так что $y_k(y)$ вычисляются с высокой точностью, но от шага число найденных смен знака конечно, зависит: при k=100 число всех нулей функции $y_k(y)$ равно 2^{100-1}=6.3383e29. Итак, неподвижную точку inf отображения F следует рассматривать как неустойчивую, так как любая ее окрестность с разрезом Re(z)=2.5 в комплексной плоскости с ростом k "уходит" от нее, а из этого разреза все больше точек попадает в нее, и эти ее дискретные прообразы все плотнее (в пределе всюду плотно) заполняют этот разрез, но все их множество, поскольку оно счетно, имеет меру нуль. Отсюда следует, что на прямой Re(z)=2.5 почти всюду не существует теоретического предела итераций. Из-за ошибок округления, хотя они и малы, прообразы inf почти никогда не попадают в inf при вычислениях и потому ведут себя так же (хаотично), как и не прообразы.

Заметим, что порядок действий в F можно переставить, т. е. записи: $F(x)=x/2+1.25+0.25/(2x-5)$ и $F(x)=(x^2-6)/(2x-5)$ – эквивалентны. Тогда выполнив команды получим тот же результат:

```
TF='xt/2+1.25+.25./(2*xt-5)';
x=-10:.5:10;      [X,Y]=meshgrid(x);         xt=X+i*Y;
n=100; for k=1:n,xt=eval(TF);end, plot(xt,'.')
z=xt(:);      [sum(abs(z-2)<.1),     sum(abs(z-3)<.1),
sum(abs(real(z)-2.5)<.1)]
```

Но команды дадут только устойчивые неподвижные точки отображения F и, как и раньше, три точки NaN, так и для такой TF мы не увидели бы такой картины:

```
TF='(xt.^2-6)./(2*xt-5)';
x=-10:.5:10; [X,Y]=meshgrid(x);   xt=X+i*Y;    n=100;
for k=1:n,xt=eval(TF);end, plot(xt,'.')
z=xt(:,26); [sum(isinf(z)), sum(isnan(z))]
```

Выполним с n=100 увидим, как все происходит до конца:

```
x=-10:.5:10;   [X,Y]=meshgrid(x);   xt=X+i*Y;   n=100;
for k=1:n, xt=eval(TF); plot(xt,'.'), pause(0), end
```

Упражнение.

1. Взяв в примере 1, некоторый угловой интервал в один градус и подобрав для него интервал по радиусу, проверить, что тогда граница области сходимости снова будет иметь сложную структуру. Такая бесконечно

сложная структура границ областей сходимости рассматривается в теории фракталов.

2. Считая итерации сходящимися, запишем уравнение $f(x)\equiv x^2-5x+6=0$ в виде: $x_{k+1}\cdot x_k-5\cdot x_{k+1}+6=0$, откуда $x_{k+1}=-6/(x_k-5)$ или $y=F(x)$, где $F(x)=-6/(x-5)$, откуда $F'(x)=6/(x-5)^2$. Поэтому $x_1=2$ – устойчивая точка, в которую итерациями стягивается вся комплексная плоскость с выколотой неустойчивой точкой $x_2=3$. Других неподвижных точек нет, хотя и не везде $|F'(x)|<1$. Убедитесь в этом аналогично примеру 2.

Лабораторная работа № 7.

Тема: Системы линейных алгебраических уравнений.

MATLAB имеет несколько команд для решения таких задач. Рассмотрим – оператор \ (деление слева). Для чисел $a\backslash b=a^{-1}b$ в отличие от $a/b=a\cdot b^{-1}$. Для невырожденной квадратной матрицы A и столбца f столбец $u=A\backslash f$ есть решение системы линейных уравнений $Au=f$, т. е. в записи по аналогии с числами $u=A^{-1}f$, но матрица A^{-1} не вычисляется, а система $Au=f$ решается методом исключения Гаусса. Вектор f может быть и матрицей, тогда каждый столбец матрицы $u=A\backslash f$ есть решение системы для соответствующего столбца из f, т. е. команда $A\backslash f$ сразу решает много систем, а матрица A будет разлагаться на множители методом Гаусса только один раз. Если матрица A не квадратная, вектор $u=A\backslash f$ есть решение системы $Au=f$ в смысле метода наименьших квадратов.

Пример 1. Решить систему $Au=f$,

```
x=1:.1:5;        m=length(x);        A=toeplitz(exp(x));
ut=sin(x)'; f=A*ut; u=A\f;
```

Здесь задается вектор-строка x=1..5 из 41 элемента, затем по вектору e^x функция toeplitz вычисляет квадратную матрицу A (41×41), далее в виде вектора-столбца задается точное решение u_t, по которому строится f, а затем решение системы $Au=f$. На графиках показан вид матрицы A:

```
mesh(A) plot(A(1,:)) plot(A(20,:)) plot(A(41,:))
```

Командами можно увидеть, что f сильно отличается от u, а точное и численное решения графически совпадают (u фиолетовая линия накрыла желтую линию u_t):

```
plot(ut) plot(f) plot([ut,u])
```

Иногда нужно сравнивать сильно разномасштабные кривые (u и f). Для этого проведем нормировку каждой из них на свой абсолютный максимум. График изображает динамику изменения кривых u и f относительно друг друга:

```
plot([u/max(abs(u)),f/max(abs(f))])
```

Пример 2. Команда det вычисляет определитель:

```
max(abs(u-ut)), det(A)
```

Откуда, u и u_t совпадают примерно в 11 знаках, хотя определитель $\|A\|$ системы очень мал. Однако, решим нашу систему по правилу Крамера, обозначив решение через u_c:

```
d=det(A);        uc=zeros(m,1);        for        k=1:m,
uc(k)=det([A(:,1:k-1),f,      A(:,k+1:m)])/d;      end,
max(abs(uc-ut)), plot([ut,uc])
```

Проверка: при k=1 A(:,1:k-1)=[], а при $k=m$ A(:,k+1:m), т. к. 1:0=[] и m+1:m=[], т. е. все матрицы C_k=[A(:,1:k-1), f, A(:,k+1:m)], k=1..m, верны. На

графике u_c снова совпадает с точным решением u_t, а погрешность почти не изменилась.

Пример 3. Трудности численного решения системы связаны с возможной близостью к нулю собственных значений матрицы A, которые определяются как все решения λ_1, …, λ_m характеристического уравнения $||A-\lambda I||=0$. MATLAB использует команду eig для нахождения собственных значений:

```
sp=eig(A);        plot(sp),        [y,yi]=min(abs(sp)),
[z,zi]=max(abs(sp)), c=z/y
```

На графике видно, что все собственные значения вещественны и никак не упорядочены: ближе всего к нулю λ_3=.1360, дальше всего λ_{41}=898.1453, модуль их отношения $c(A)\equiv c$=6.6040e+3 называется *числом обусловленности* (condition number) матрицы A. При обращении [V,D]=eig(A) в диагональной матрице D расположены λ_k (раньше они были в столбце s_p), а в матрице V в виде столбцов – соответствующие им собственные векторы v_k с единичной среднеквадратичной нормой $\sum\sqrt{}(v_k^2)=1$, k=1..m. Команда eig имеет неплохую точность при решении спектральных задач средней сложности. В задаче есть λ_k относительно близкие друг к другу; проверим, что это так:

```
ssp=sort(sp);          v=1:m-1;          di=diff(ssp);
min(abs([di./ssp(v);di./ssp(v+1)]))
```

Нормируем разности для каждой пары элементов; ответ даст: нашлась пара, у которой совпадает более двух знаков, т. е. наш пример в спектральном плане не тривиальный. Далее, сделаем матрицу A вырожденной с помощью дублирования ее строк и посмотрим, как это скажется на результатах:

```
r=zeros(1,m-1);   for   k=1:m-1,   v=[1:k,k,k+2:m];
C=A(v,:); r(k)=min(abs(eig(C))); end, plot(r)
```

Результаты неплохие – r_{max}=1e-14, есть даже чистый ноль. Рассмотрим собственные векторы A:

```
[V,D]=eig(A);          D=V'*V;          D(1:m+1:m^2)=0;
mcv=max(abs(D(:)))
```

Ответ 7.7542e-16 означает, что собственные векторы с высокой степенью точности ортогональны, как и должно быть для симметричной матрицы A.™

Теоретическая оценка погрешности при решении линейных систем. Напомним смысл числа обусловленности $c(A)$. Пусть $Au=f\neq 0$, $A\circledR u=\circledR f\neq 0$, $\mathrm{Re}(f)=\max(|\circledR f|)/\max(|f|)$, тогда $u\neq 0$, $\circledR u\neq 0$, т. к. A невырожденная. Будем рассматривать $\circledR f$ как ошибку в f (ввиду линейности задачи, считать вектор $\circledR f$ малым не нужно). Оценим величину: $\mathrm{Re}(u)=\max(|\circledR u|)/\max(|u|)$ для всех допустимых f и $\circledR f$, т. е. относительную ошибку, являющуюся весьма общей характеристикой матрицы. Если y, z, c – числа, определенные выше, то $\max(|\circledR u|)\le z^{-1}\max(|\circledR f|)$, ибо $\circledR u=A^{-1}u$ и z^{-1} – максимальное по модулю собственное значение для A^{-1}, причем равенство для $\max(|\circledR u|)$ теоретически возможно. Аналогично, $\max(|u|)\ge y^{-1}\max(|f|)$, ибо y^{-1} – минимальное по модулю

собственное значение для A^{-1}, и знак равенства здесь также возможен. Объединяя оценки для числителя и знаменателя Re(u), имеем Re(u) ≤ z^{-1}/y^{-1}(max(|®f |)/max(|f|)), или Re(u) ≤ (y/z) Re(f), т. е. Re(u) ≤ c·Re(f), причем равенство обязательно достигается хотя бы для одной пары f и ®f. Другими словами, c(A) – sup роста относительной ошибки при решении системы: если относительная ошибка правой части равна Re(f), то относительная ошибка решения Re(u) не превзойдет ее более чем в c(A) раз. Таким образом, ®f переходит в ®u с возрастанием компонент не более чем в z^{-1} раз, а f переходит в u с уменьшением компонент не более чем в y^{-1} раз, так что Re(u) ≤(z^{-1}/y^{-1})Re(f). Все сказанное для c(A) справедливо и для среднеквадратичной нормы: ||f||=∑ √ |f^2|.

Команда cond(A) вычисляет значение c(A) для квадратной невырожденной матрицы, но в cond спектр sp находится не для A, а для B=A'A, ибо тогда вся спектральная задача для B становится вещественной, а все ее собственные векторы взаимно ортогональны, что существенно уменьшает среднеквадратичную ошибку при этих таких вычислениях. Для получения y и z нужно лишь извлечь квадратный корень из y(B) и z(B) – это также делается командой: **cond(A)**. В нашем примере c(A)=6.6040e+3, что совпадает со значением полученным ранее, так как A – симметричная матрица.

Практическая оценка погрешности. Число обусловленности может быть не всегда подходящим для оценки погрешности. Это может случится как при больших n, так и том случае, когда ошибка ®f специфически неравномерна и имеет характерный профиль – тогда желательно иметь и профиль для поточечных ошибок ®u_k, чего cond(A) дать не может. Тогда приходится прибегать к непосредственному моделированию ошибок, которое состоит в задании некоторого множества случайных возмущений правой части и решении всех таких систем с последующим поточечным описанием границ получившихся решений. Проиллюстрируем этот способ оценки ошибки на нашем примере:

```
hx=.1;  x=1:hx:5;  m=length(x);  A=toeplitz(exp(x));
ut=sin(x)'; f=A*ut; u=A\f;
```

Пусть ошибка ®f(x) в f(x) есть равномерно распределенная случайная величина, по модулю не превосходящая значения g(x)=1e-4*∫f(t) ®t в пределах от 1 до x. Далее, приведем в исходное состояние счетчик случайных чисел, зададим профиль g=g(x) максимально допустимой ошибки в точке x, число возмущений n правой части f, матрицу V m×n из продублированного n раз столбца (1:m)'. На графике видно, что ошибка больше там, где |f| меньше:

```
g=1e-4*hx*abs(cumsum(f));   rand('state',0);   n=30;
V=(1:m)'*ones(1,n);   plot([f/max(abs(f)),g/max(g)]),
grid
```

Создадим матрицу F возмущенных правых частей из m строк и n столбцов путем прибавления к размноженному n раз вектору f возмущений в нужных

границах; решим систему $AU=F$ и построим график решений U и возмущений вместе с их границей:

```
F=f(V)+g(V).*(2*rand(m,n)-1);  U=A\F;plot(U),  pause,
plot([g(V).*(2*rand(m,n)-1),g])
```

А затем путем обработки U вдоль строк найдем поточечные границы решений (u_a – верхняя, u_i – нижняя) построим график точного решения и этих границ:

```
ua=max(U,[],2); ui=min(U,[],2); plot([ui,ut,ua])
```

Результат плохой, ибо на графике **plot(F)** возмущений не видно (масштаб f убил их), а в U они отразились слишком сильно, но формально он не противоречит оценке, связанной с числом обусловленности $c(A)$. Действительно, без учета профиля ®f максимальная ошибка есть:

```
sp=eig(A); me=min(abs(sp))^(-1)*max(g)
```

Отсюда убедимся, что величина ее не превосходит, что свидетельствует о формальной согласованности обоих методов оценки погрешности:

```
max([ua-ut;ut-ui])
```

Еще раз, применим критерий cond(A) и для среднеквадратичной нормы, для чего нам нужно пройтись по всем столбцам k=1..n матриц U и F и вычислить $\max\{(\|®u\|/\|u\|) / (\|®f\|/\|f\|))\}$:

```
max((sum((U-ut(V)).^2).^.5./sum(U.^2).^.5)./(sum((F-
f(V)).^2).^.5./sum(F.^2).^.5))
```

Получим результат меньший, чем $c(A)$ – нет противоречия. На графике все линии уже практически совпадут:

```
gm=max(g); plot(A\[f-gm,f,f+gm])
```

Таким способом можно моделировать ошибку, если преобразование A^{-1} монотонно, но у нас это не так, и поэтому получается такой разброс в U:

```
gm=100*max(g);plot(A\[f-gm,f,f+gm])
```

С помощью такого приема можно выяснить монотонно ли преобразование $y=Q(x)$, что не всегда удается определить теоретически: если все отклики $Y=Q(X)$, $x–a<X<x+a$, $a>0$, лежат между $Q(x–a)$ и $Q(x+a)$, то преобразование Q монотонно, и надо лишь взять значение a таким, чтобы результат был виден на графике.

Рассмотрим, как изменятся результаты задачи при увеличении m – порядка матрицы A:

```
clear  all,  hx=.01;  x=1:hx:5;  A=toeplitz(exp(x));
ut=sin(x)'; f=A*ut;u=A\f; plot([ut,u]), c=cond(A)
```

Здесь шаг уменьшен в 10 раз, так что теперь порядок m=401 – довольно высокий; $c(A)$=6.0804e5 возросло почти в 100 раз, т.е. обусловленность ухудшилась, однако величина еще достаточно мала:

```
max(abs(u-ut))
```

Такое расхождение с теорией говорит о том, что даже при сохранении смысла задачи увеличение ее размерности не позволяет автоматически

применять критерий числа обусловленности к оценке ошибок округления, т. е. надо внимательно относится к выбору числа m. Определим собственные векторы A:

```
[V,D]=eig(A);  D=V'*V;  m=length(x);  D(1:m+1:m^2)=0;
mcv=max(abs(D(:)))
```

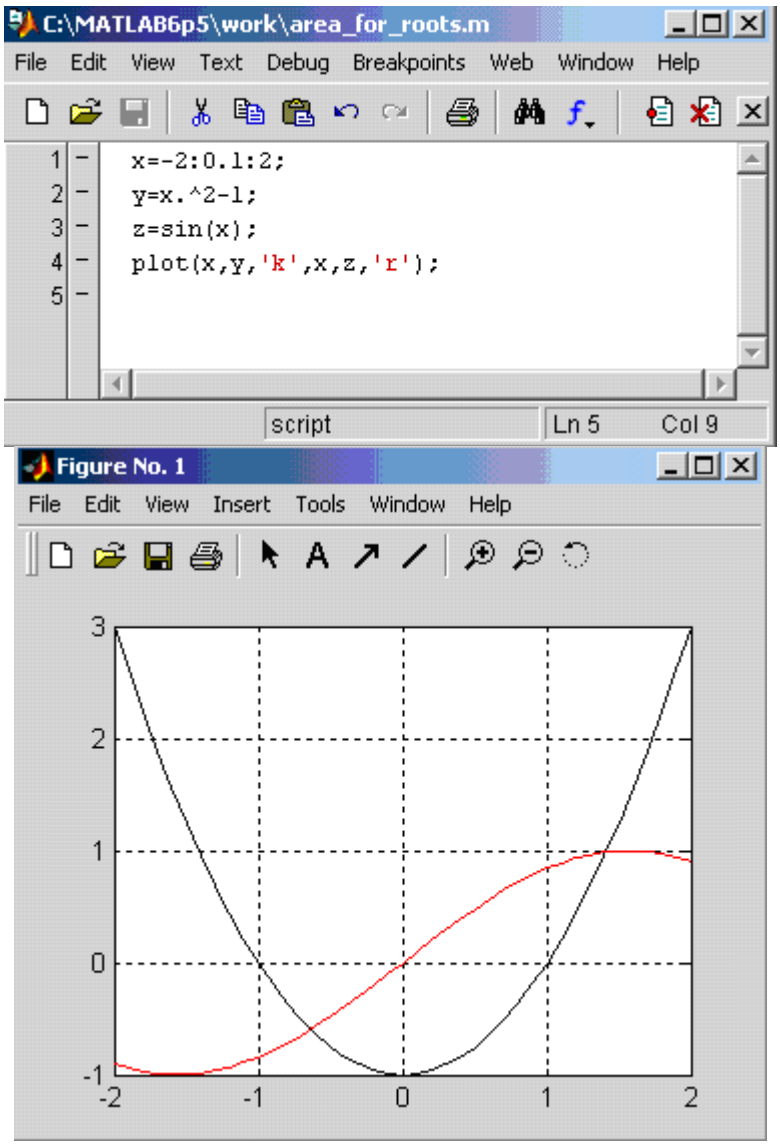
Ответ свидетельствует о высокой степени ортогональности, жордановы клетки порядка выше первого могут быть тогда, когда $mcv(A)>0.99$.

Лабораторная работа № 8.

Тема: Решение нелинейных уравнений.

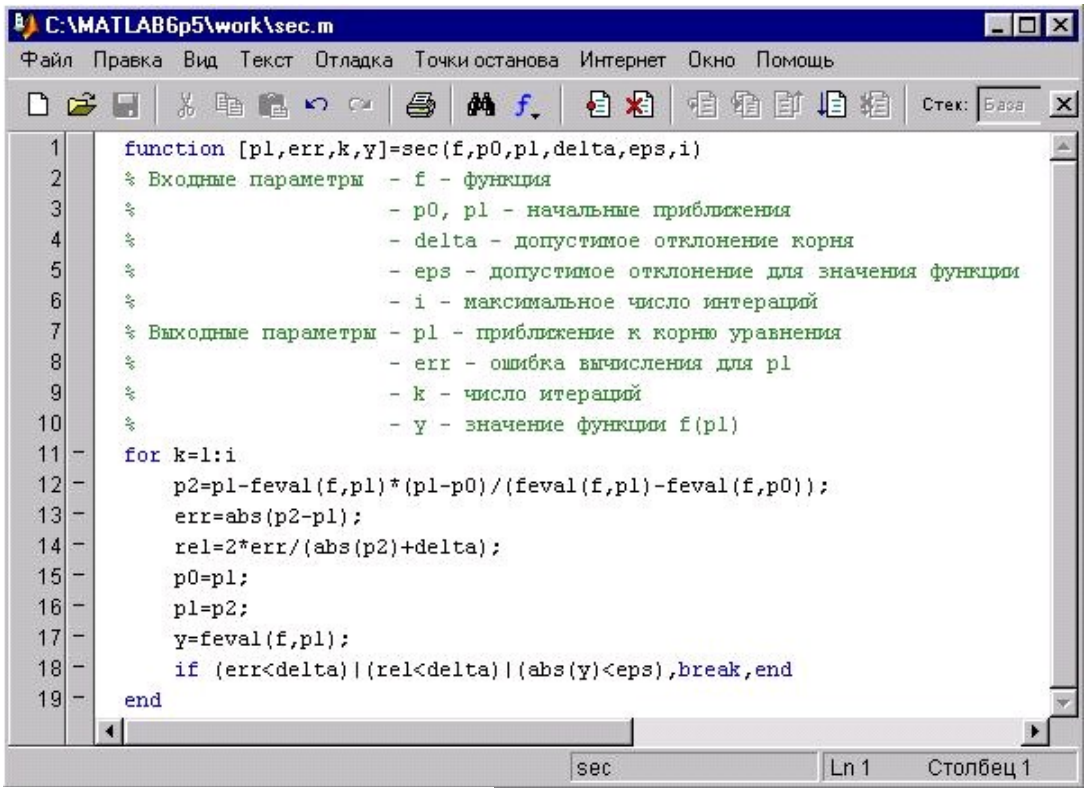
Пример 1. Найти промежутки локализации корней уравнения $x^2 - \sin x - 1 = 0$.

Представив уравнение в виде $x^2 - 1 = \sin x$, построим графики функций $y_1 = x^2 - 1$ и $y_2 = \sin x$, как показано на рисунке. Совместное рассмотрение графиков позволяет сделать заключение, что данное уравнение имеет два корня: $\xi_1 \in [-1,0]$ и $\xi_2 \in [1,\pi]$.



Пример 2. Методом хорд найти корень уравнения $f(x) = x^3 + 8x^2 - 14x - 20 = 0$, расположенного на отрезке $[2,3]$.

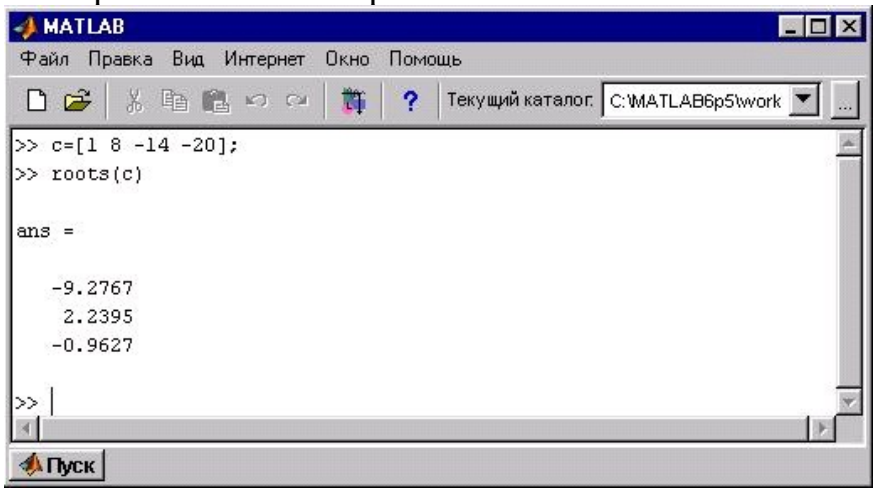
Реализация примера с использованием Matlab представлена на рисунке.




```
>> [p1,err,k,y]=sec('f',2,3,10e-4,10e-5,10)
p1  = 2.23944936855661
err = 3.852721542894066e-004
k   = 4
y   = -8.587285155670088e-005
>>
```

Пример 3. Найти корень уравнения $x^3 + 8x^2 - 14x - 20 = 0$, используя функцию roots().

Для нахождения корней полинома в пакете Matlab предусмотрена соответствующая функция **roots()**, возвращающая вектор-столбец, компоненты которого являются корнями полинома.



Индивидуальные задания

1. Используя один из методов, найти область существования решения предложенного уравнения.

2. Найти корень уравнения $f(x) = 0$ в найденной области с точностью $\varepsilon = 10^{-4}$ указанными методами в соответствии с номером варианта. Реализовать алгоритмы указанных методов.
3. Определить число итераций каждого метода и оценить погрешности решения.
4. Выполнить проверку, используя возможности пакета Matlab. Сравнить решения, полученные всеми способами. Сделать вывод.

Номер варианта	**Уравнение** $f(x) = 0$	**Численные методы**
1	$x^3 + 0.3 \cdot x^2 + 1.8 \cdot x - 15 = 0$	Метод Ньютона Метод Чебышева
2	$x^3 + 0.2 \cdot x^2 + 5 \cdot x + 4 = 0$	Метод половинного деления Двушаговый метод Ньютона
3	$x^3 + 0.7 \cdot x^2 + 4.7 \cdot x - 1.5 = 0$	Метод секущих Метод Чебышева
4	$x^3 + 11 \cdot x^2 - 8 \cdot x - 25 = 0$	Упрощенный метод Ньютона Метод секущих
5	$x^3 + 3 \cdot x^2 + 3 \cdot x + 4 = 0$	Метод половинного деления Метод Стеффенсена
6	$x^3 - 2 \cdot x - 0.4 = 0$	Метод половинного деления Метод Чебышева
7	$x^3 + x^2 + 3 \cdot x - 13 = 0$	Разностный метод Ньютона Метод секущих
8	$x^3 + x^2 + 3.8 \cdot x - 2 = 0$	Метод половинного деления Двушаговый метод Ньютона
9	$x^3 + 3.5x^2 + 12 \cdot x + 19 = 0$	Упрощенный метод Ньютона Метод Чебышева
10	$x^3 + 6x^2 - 17 \cdot x - 21 = 0$	Метод Ньютона-Шредера Метод хорд

Лабораторная работа № 9.

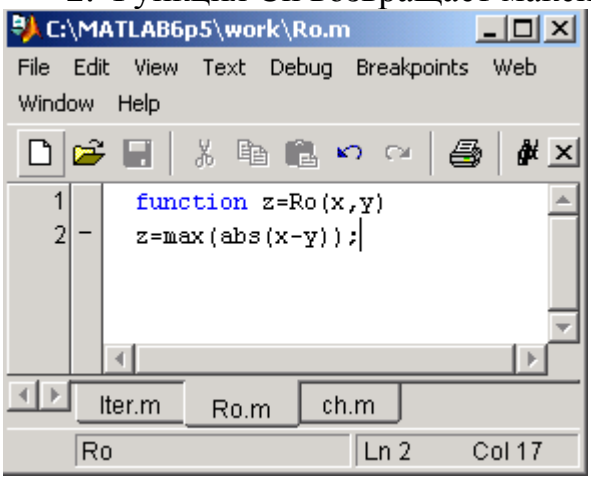
Тема: Решение систем линейных алгебраических уравнений.

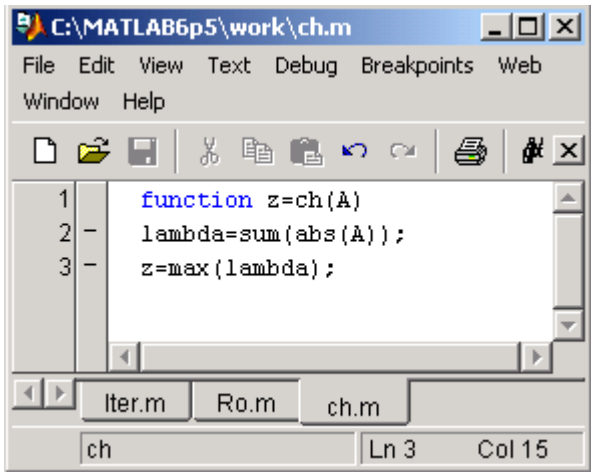
Пример 1. Методом простой итерации найти решение системы уравнений:

$$\begin{cases} x_1 = 0.12x_1 - 0.23x_2 + 0.25x_3 - 0.16x_4 + 1.24, \\ x_2 = 0.14x_1 + 0.34x_2 - 0.18x_3 + 0.24x_4 - 0.89, \\ x_3 = 0.13x_1 + 0.03x_2 + 0.46x_3 - 0.32x_4 + 1.15, \\ x_4 = 0.12x_1 - 0.05x_2 + 0.15x_4 - 0.57 \end{cases}$$ с точность $\varepsilon = 10^{-3}$

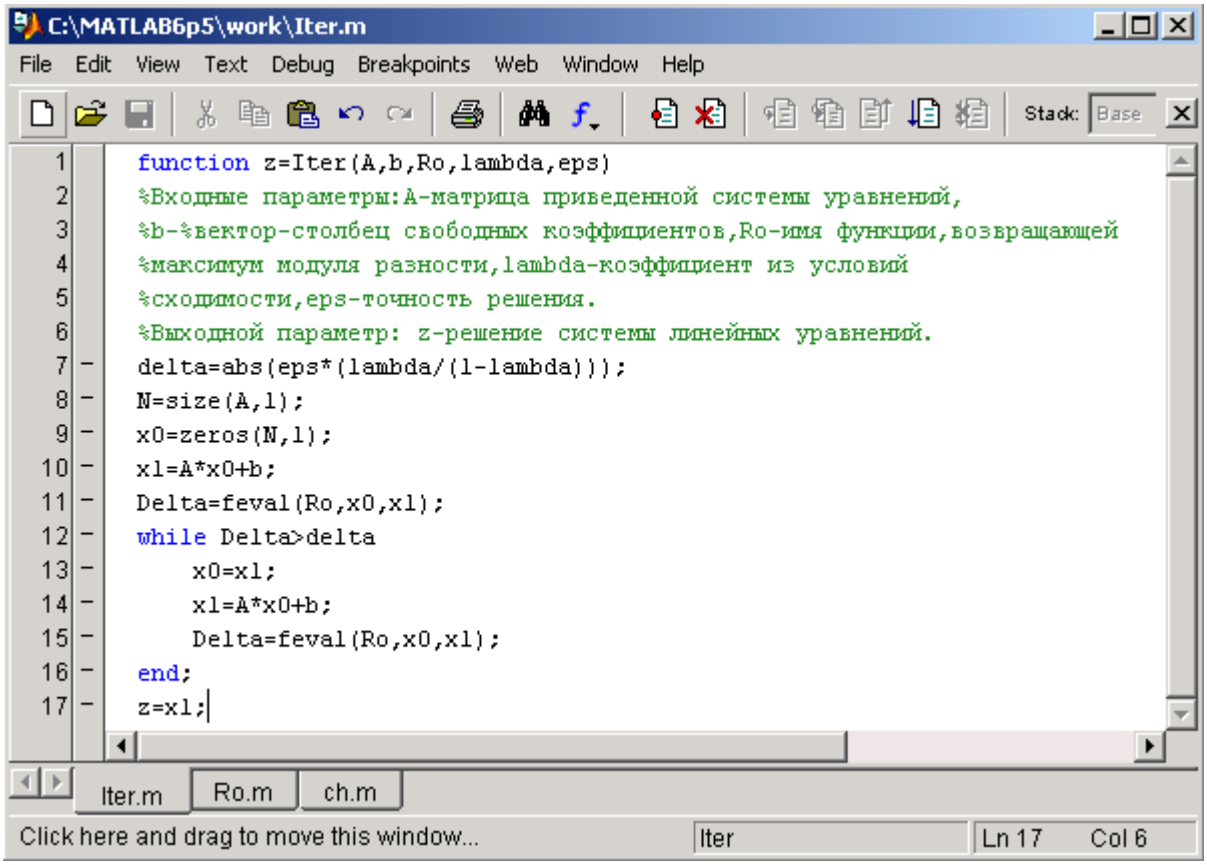
1. Функция Ro возвращает значение расстояния между точками в пространстве с метрикой $\rho(x, y) = \max\limits_{i=1,n} |x_i - y_i|$.

2. Функция Ch возвращает максимальное значение λ





3. Функция Iter возвращает решение системы линейных уравнений полученное с заданной точностью ε .



Результат работы программы, определяющей решение системы линейных уравнений методом простой итерации:

```
>> A=[0.12,-0.23,0.25,-0.16;0.14,0.34,-0.18,0.24;0.13,0.03,0.46,-0.32;0.12,-0.05,0,0.15]

A =  0.1200   -0.2300    0.2500   -0.1600
     0.1400    0.3400   -0.1800    0.2400
     0.1300    0.0300    0.4600   -0.3200
     0.1200   -0.0500         0    0.1500

>> b=[1.24;-0.89;1.15;-0.57]

b =  1.2400
    -0.8900
     1.1500
    -0.5700

>> lambda=ch(A)

 lambda = 0.8900

>> x=Iter(A,b,'Ro',lambda,10e-3)

 x = 2.6633
    -1.6192
     2.7984
    -0.1995
```

Для решения систем линейных уравнений средствами пакета Matlab применяются операторы возведения в степень и умножения, действие которых определяется правилами линейной алгебры.

Пример 2. Используя встроенные функции пакета Matlab, найдите решение исходной системы линейных уравнений.

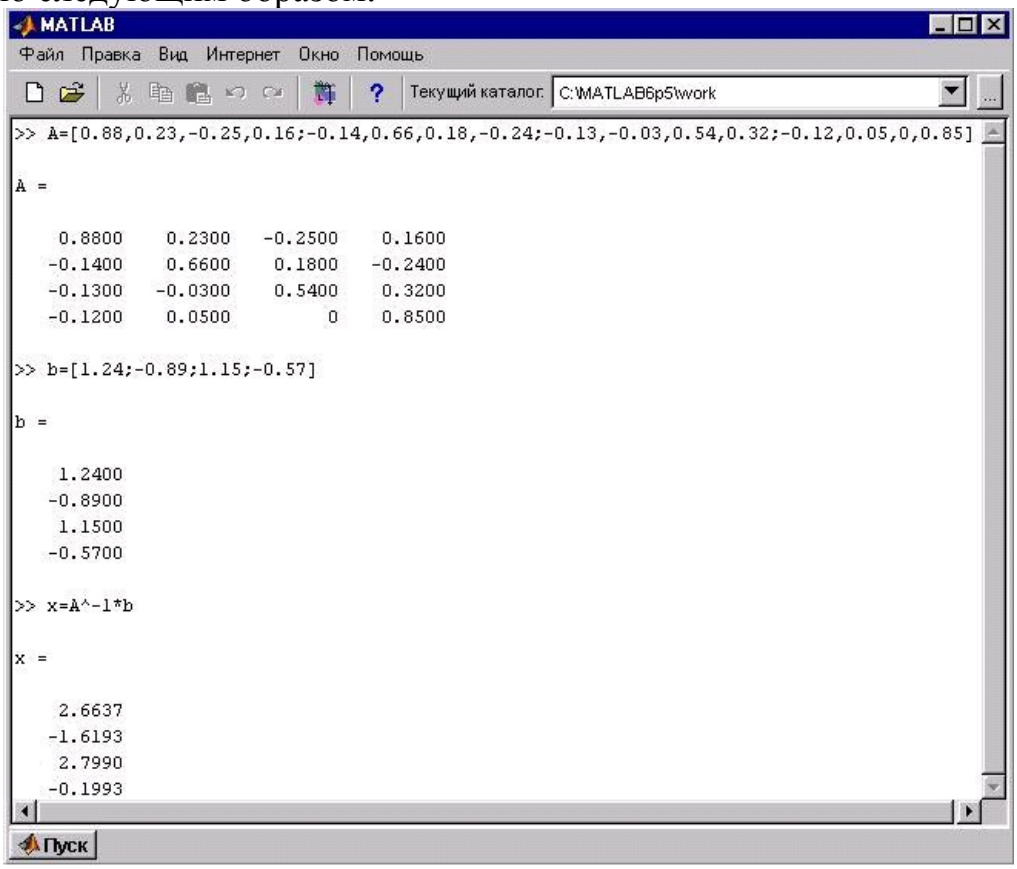
Приведем систему к обычному виду

$$\begin{cases} x_1 - 0.12x_1 + 0.23x_2 - 0.25x_3 + 0.16x_4 = 1.24, \\ x_2 - 0.14x_1 - 0.34x_2 + 0.18x_3 - 0.24x_4 = -0.89, \\ x_3 - 0.13x_1 - 0.03x_2 - 0.46x_3 + 0.32x_4 = 1.15, \\ x_4 - 0.12x_1 + 0.05x_2 - 0.15x_4 = -0.57, \end{cases}$$

или

$$\begin{cases} 0.88x_1 + 0.23x_2 - 0.25x_3 + 0.16x_4 = 1.24, \\ -0.14x_1 + 0.66x_2 + 0.18x_3 - 0.24x_4 = -0.89, \\ -0.13x_1 - 0.03x_2 + 0.54x_3 + 0.32x_4 = 1.15, \\ -0.12x_1 + 0.05x_2 + 0.85x_4 = -0.57 \end{cases}$$

Решение последней системы средствами пакета Matlab может быть записано следующим образом:



Индивидуальные задания

1. Реализовать алгоритм метода Гаусса с выбором главного элемента (по столбцу или по строке) в пакете Matlab и найти решение предложенной системы уравнений $AX = B$. Значения матрицы A и вектора B приведены в таблице.

2. Привести систему уравнений к виду, пригодному для итерационного процесса Зейделя, реализовать метод в пакете Matlab и решить систему уравнений с точностью $\varepsilon = 10^{-3}$ и исходными данными, приведенными в таблице.

Номер варианта	Значения матрицы A и вектора B

1	$A = \begin{pmatrix} 2.1 & -4.5 & -2 \\ 3 & 2.5 & 4.3 \\ -6 & 3.5 & 2.5 \end{pmatrix}, B = \begin{pmatrix} 19.07 \\ 3.21 \\ -18.25 \end{pmatrix}$
2	$A = \begin{pmatrix} -6.7 & 5.9 & 4.1 & -9.1 \\ 5.4 & -8.7 & 11.2 & -0.5 \\ 2.7 & -4.9 & 6.8 & 3.3 \\ 1.6 & 7.2 & 8.3 & -7.9 \end{pmatrix}, B = \begin{pmatrix} 7.12 \\ 58.07 \\ 47.1 \\ 36.8 \end{pmatrix}$
3	$A = \begin{pmatrix} 3.81 & 0.25 & 1.28 & 0.75 \\ 2.25 & 1.32 & 4.58 & 0.49 \\ 5.31 & 6.28 & 0.98 & 1.04 \\ 9.39 & 2.45 & 3.35 & 2.28 \end{pmatrix}, B = \begin{pmatrix} 4.21 \\ 6.47 \\ 2.38 \\ 10.48 \end{pmatrix}$
4	$A = \begin{pmatrix} 2.34 & -4.21 & -11.61 \\ 8.04 & 5.22 & 0.27 \\ 3.92 & -7.99 & 8.37 \end{pmatrix}, B = \begin{pmatrix} 14.41 \\ -6.44 \\ 55.56 \end{pmatrix}$
5	$A = \begin{pmatrix} 0.18 & 0.25 & -0.44 \\ 0.42 & -0.35 & 1.12 \\ 1.14 & 0.12 & -0.83 \end{pmatrix}, B = \begin{pmatrix} 1.15 \\ 0.86 \\ 0.68 \end{pmatrix}$
6	$A = \begin{pmatrix} 21.54 & -95.5 & -96.12 \\ 10.22 & -91.06 & -7.34 \\ 51.21 & 12.29 & 86.45 \end{pmatrix}, B = \begin{pmatrix} -49.93 \\ -12.46 \\ 60.81 \end{pmatrix}$
7	$A = \begin{pmatrix} 2.51 & -3.12 & 4.64 \\ -3.25 & 2.62 & 1.85 \\ -6.53 & -3.5 & 7.31 \end{pmatrix}, b = \begin{pmatrix} -1.05 \\ -14.46 \\ -17.73 \end{pmatrix}$
8	$A = \begin{pmatrix} 8.2 & 1.4 & -2.3 & 0.2 \\ -1.6 & 5.4 & -7.7 & 3.1 \\ 0.7 & 1.9 & -8.5 & 4.8 \\ 5.3 & -5.9 & 2.7 & -7.9 \end{pmatrix}, b = \begin{pmatrix} 32.76 \\ 54.39 \\ 59.18 \\ -71.95 \end{pmatrix}$
9	$A = \begin{pmatrix} 4.21 & 22.42 & 3.85 \\ 2.31 & 31.49 & 1.52 \\ 3.49 & 4.85 & 28.72 \end{pmatrix}, B = \begin{pmatrix} 30.24 \\ 4.095 \\ 42.81 \end{pmatrix}$
10	$A = \begin{pmatrix} 0.63 & 0.07 & 0.2 \\ 0.05 & 0.34 & 0.3 \\ 0.15 & 0.1 & 0.71 \end{pmatrix}, B = \begin{pmatrix} 0.34 \\ 0.32 \\ 0.42 \end{pmatrix}$

3. Найти решение той же системы, используя возможности пакета Matlab. Сравните найденное решение с полученным при реализации метода

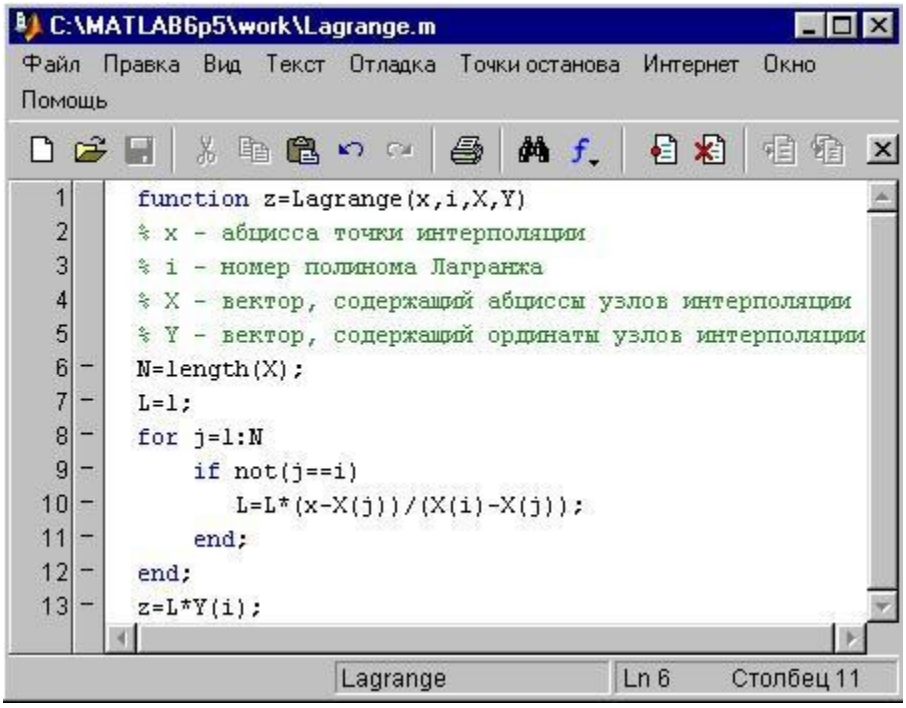
Гаусса и метода Зейделя. За сколько шагов достигается заданная точность в каждом из методов?

Лабораторная работа № 10.

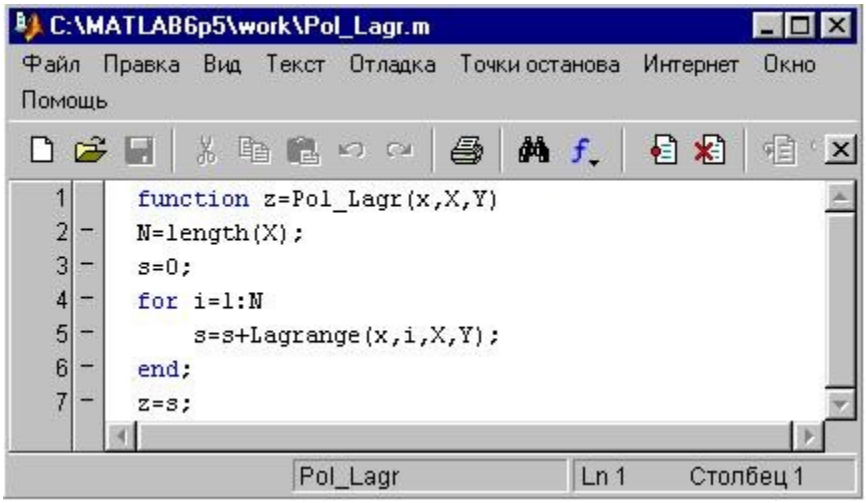
Тема: Интерполирование функций.

Пример 1. Решить задачу интерполяции с помощью многочлена Лагранжа для функции $f(x) = \sin(x)$, заданной таблично на интервале $[0, 2\pi]$ при $n = 8$.

Создадим функцию, возвращающую значение многочлена $l_i(x)$.



Создадим файл Pol_Lagr.m, возвращающий значение полинома Лагранжа.



В файле задается табличная функция, вычисляются ее точные значения, значения полинома Лагранжа.

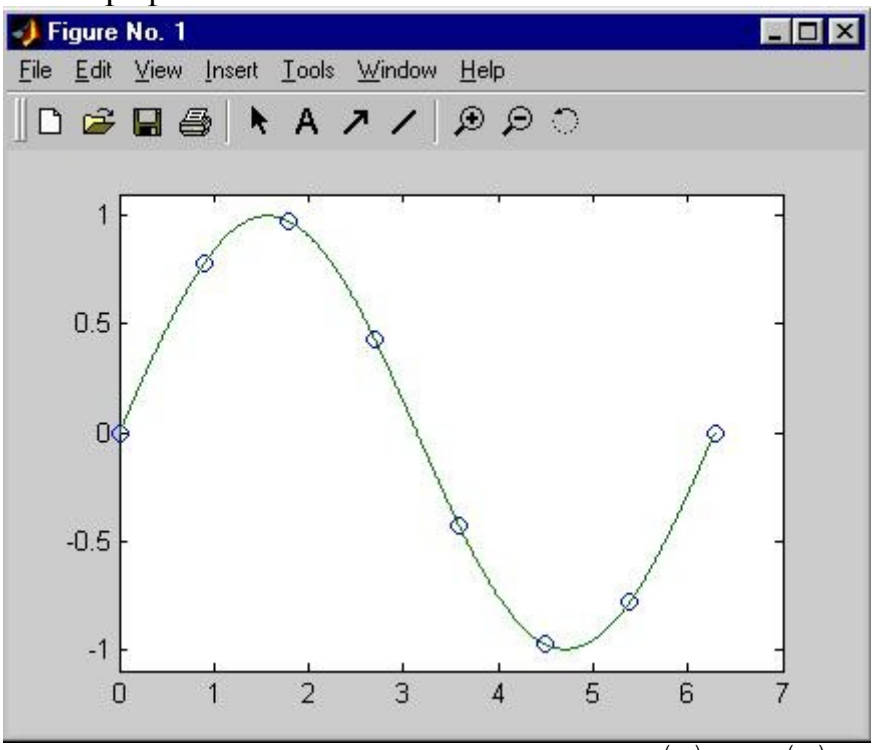
C:\MATLAB6p5\work\Lag.m

Файл Правка Вид Текст Отладка Точки останова Интернет Окно Помощь

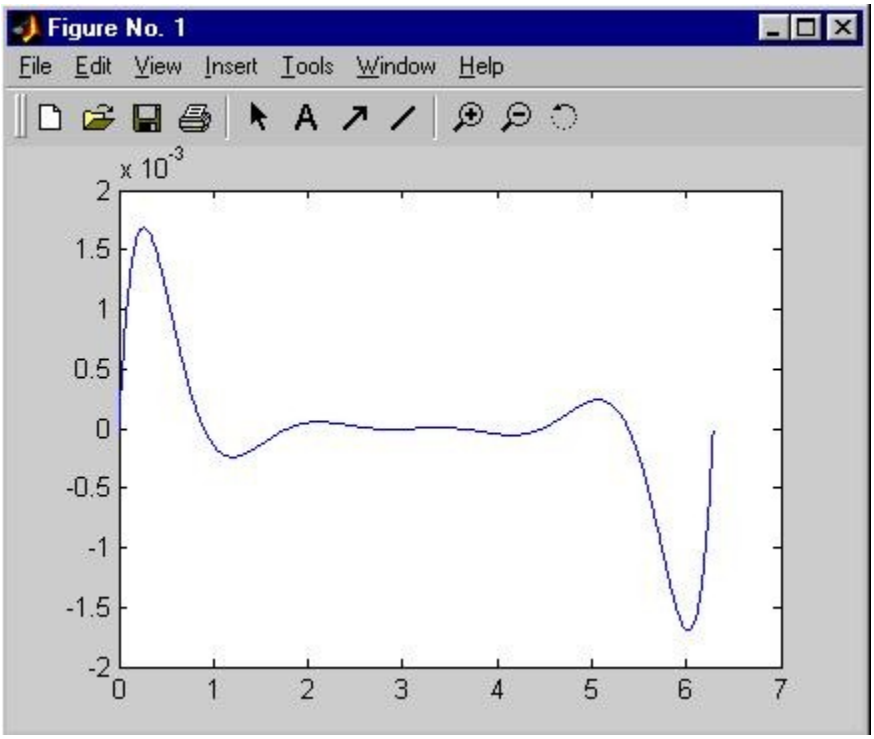
```
% Задаем табличные значения интерполируемой функции
N=8;
i=1:N;
x(i)=2*pi/(N-1)*(i-1);
y=sin(x);
% Задаем число промежуточных точек, вычисляем их координаты
% и значение интерполируемой функции
M=1000;
j=1:M;
X(j)=2*pi/(M-1)*(j-1);
Y=sin(X);
% Вычисляем значение полинома Лагранжа в промежуточных точках
for j=1:M
    Y2(j)=Pol_Lagr(X(j),x,y);
end;
plot(x,y,'o', X,Y2);
pause
plot(X,Y2-Y);
```

script Ln 12 Столбец 62

Значения табличной функции и интерполированные значения функции представлены на графике.

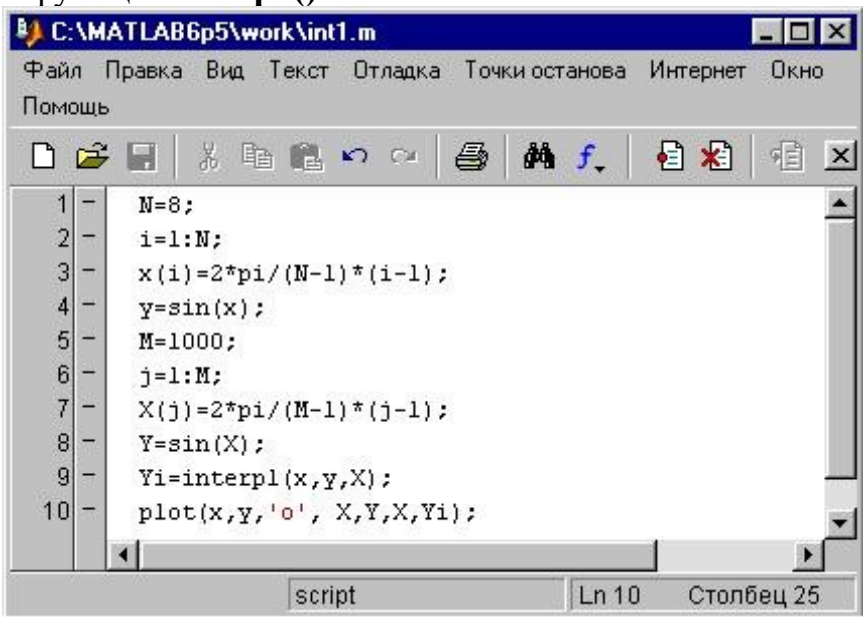


Погрешность аппроксимации функции $f(x) = \sin(x)$ полиномом Лагранжа имеет вид, изображенный на рисунке.

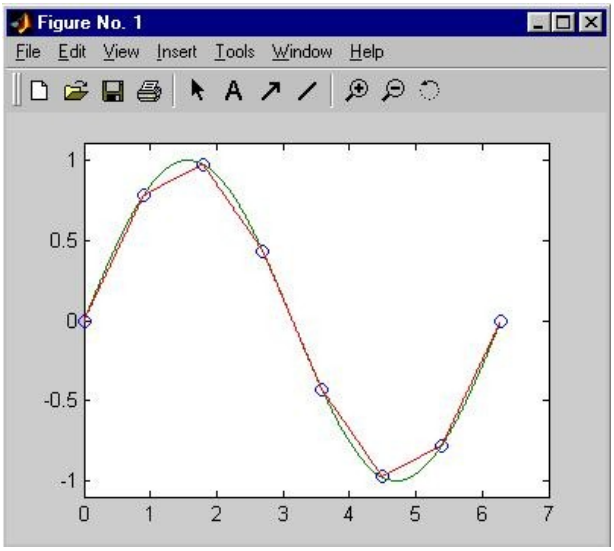


Пример 2. Решить задачу интерполяции для функции $f(x) = \sin(x)$, заданной таблично на интервале $[0, 2\pi]$ при $n = 8$ средствами пакета Matlab.

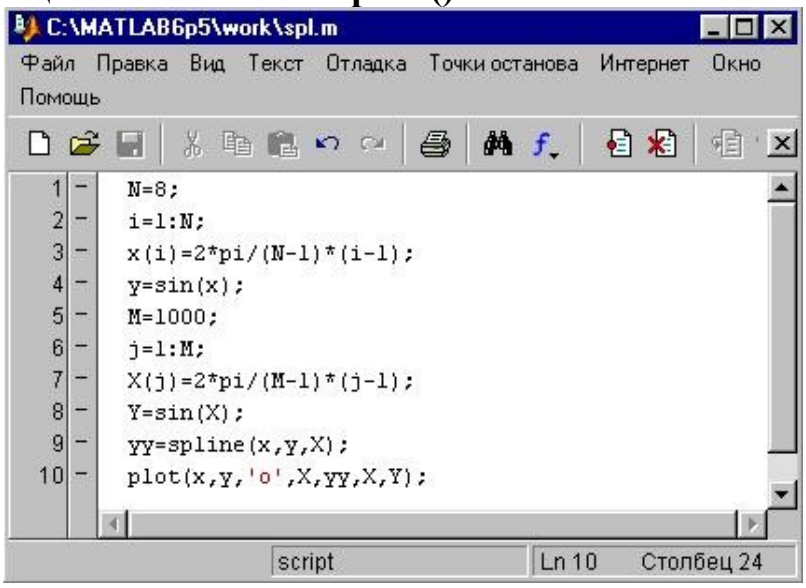
Для решения задачи одномерной интерполяции в пакете Matlab используется функция **interp1()**.



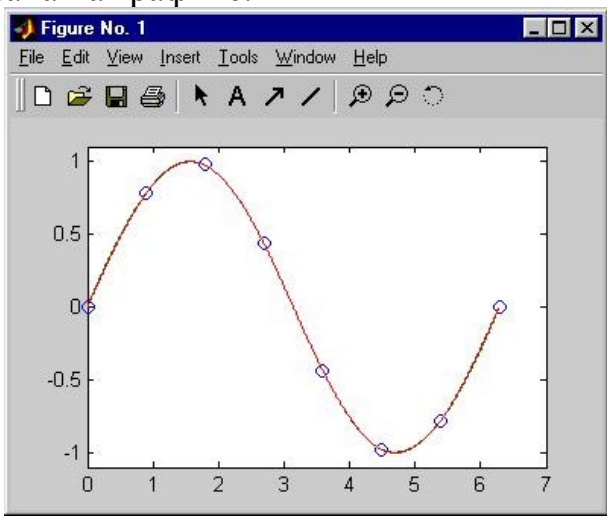
Исходные данные и результат линейной интерполяции представлены на рисунке.



Данную задачу можно решить с помощью кубических сплайнов, используя функцию пакета Matlab **spline()**.



Визуализация исходных данных и результатов кубической сплайн-интерполяции показана на графике.



Индивидуальные задания

1. Пользуясь первой и второй интерполяционными формулами Ньютона, вычислить значение функции для указанных значений аргумента.

2. Решить задачу интерполяции средствами пакета Matlab с помощью линейных и кубических сплайнов. Привести графики исходных данных, результатов сплайн-интерполяции и погрешности аппроксимации.

Номер варианта	x	$f(x)$	Значения аргумента	Номер варианта	x	f(i)	Значения аргумента
1	1.50	0.51183	x = 1.50911 x = 1.59513	6	0.50	1.6487	x = 0.50721 x = 0.56894
	1.51	0.50624			0.51	1.6653	
	1.52	0.50064			0.52	1.6820	
	1.53	0.49503			0.53	1.6989	
	1.54	0.48940			0.54	1.7160	
	1.55	0.48376			0.55	1.7333	
	1.56	0.47811			0.56	1.7507	
	1.57	0.47245			0.57	1.7683	
	1.58	0.46678			0.58	1.7860	
	1.59	0.46110			0.59	1.8040	
	1.60	0.45540			0.60	1.8221	
2	0.00	0.28081	x = 0.01928 x = 0.47113	7	1.1	0.89121	x = 1.1511 x = 2.0316
	0.05	0.31270			1.2	0.93204	
	0.10	0.34549			1.3	0.96356	
	0.15	0.37904			1.4	0.98545	
	0.20	0.41318			1.5	0.99749	
	0.25	0.44774			1.6	0.99957	
	0.30	0.48255			1.7	0.99166	
	0.35	0.51745			1.8	0.97385	
	0.40	0.55226			1.9	0.94630	
	0.45	0.58682			2.0	0.90930	
	0.50	0.62096			2.1	0.86321	

3	1.0	0.5652		8	0.10	3.63004	
	1.1	0.6375			0.35	3.75680	
	1.2	0.7147			0.60	3.88933	
	1.3	0.7973			0.85	4.03258	
	1.4	0.8861			1.10	4.19310	
	1.5	0.9817	$x = 1.0113$		1.35	4.38042	$x = 0.1006$
	1.6	1.0848	$x = 1.9592$		1.60	4.60963	$x = 2.5304$
	1.7	1.1964			1.85	4.90697	
	1.8	1.3172			2.10	5.32331	
	1.9	1.4482			2.35	5.97322	
	2.0	1.5906			2.60	7.18210	
4	0.50	1.6487		9	1.50	0.51183	
	0.51	1.6653			1.51	0.50624	
	0.52	1.6820			1.52	0.50064	
	0.53	1.6989			1.53	0.49503	
	0.54	1.7160			1.54	0.48940	
	0.55	1.7333	$x = 0.52301$		1.55	0.48376	$x = 1.50253$
	0.56	1.7507	$x = 0.58967$		1.56	0.47811	$x = 1.59614$
	0.57	1.7683			1.57	0.47245	
	0.58	1.7860			1.58	0.46678	
	0.59	1.8040			1.59	0.46110	
	0.60	1.8221			1.60	0.45540	
5	0.10	3.63004		10	0.00	0.28081	
	0.35	3.75680			0.05	0.31270	
	0.60	3.88933			0.10	0.34549	
	0.85	4.03258			0.15	0.37904	
	1.10	4.19310			0.20	0.41318	
	1.35	4.38042	$x = 0.10056$		0.25	0.44774	$x = 0.02475$
	1.60	4.60963	$x = 2.57321$		0.30	0.48255	$x = 0.48675$
	1.85	4.90697			0.35	0.51745	
	2.10	5.32331			0.40	0.55226	
	2.35	5.97322			0.45	0.58682	
	2.60	7.18210			0.50	0.62096	

Лабораторная работа № 11.

Тема: Численное дифференцирование и интегрирование.

Пример 1. Вычислить значение производной функции $f(x) = \sin(0.1 \cdot x^2)$ на отрезке $[0, 6\pi]$.

Листинг программы, численно дифференцирующей аналитически заданную функцию, представлен в следующем окне.

C:\MATLAB6p5\work\dif.m

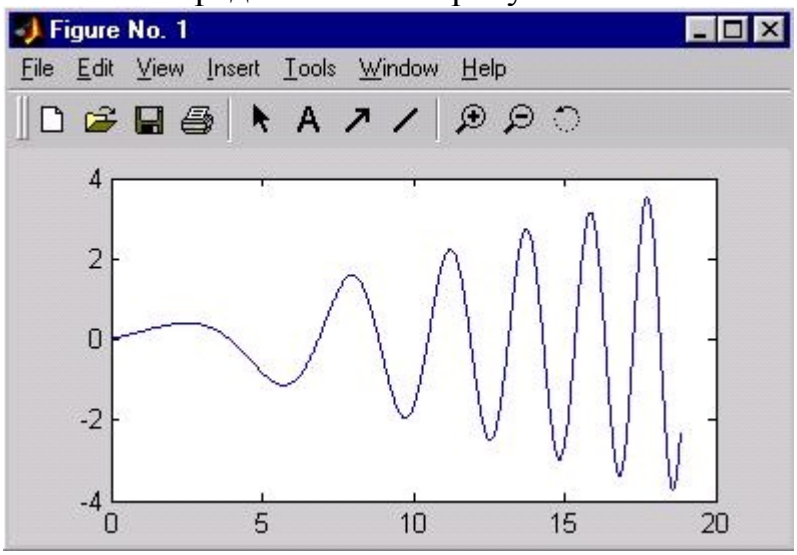
Файл Правка Вид Текст Отладка Точки останова Интернет Окно Помощь

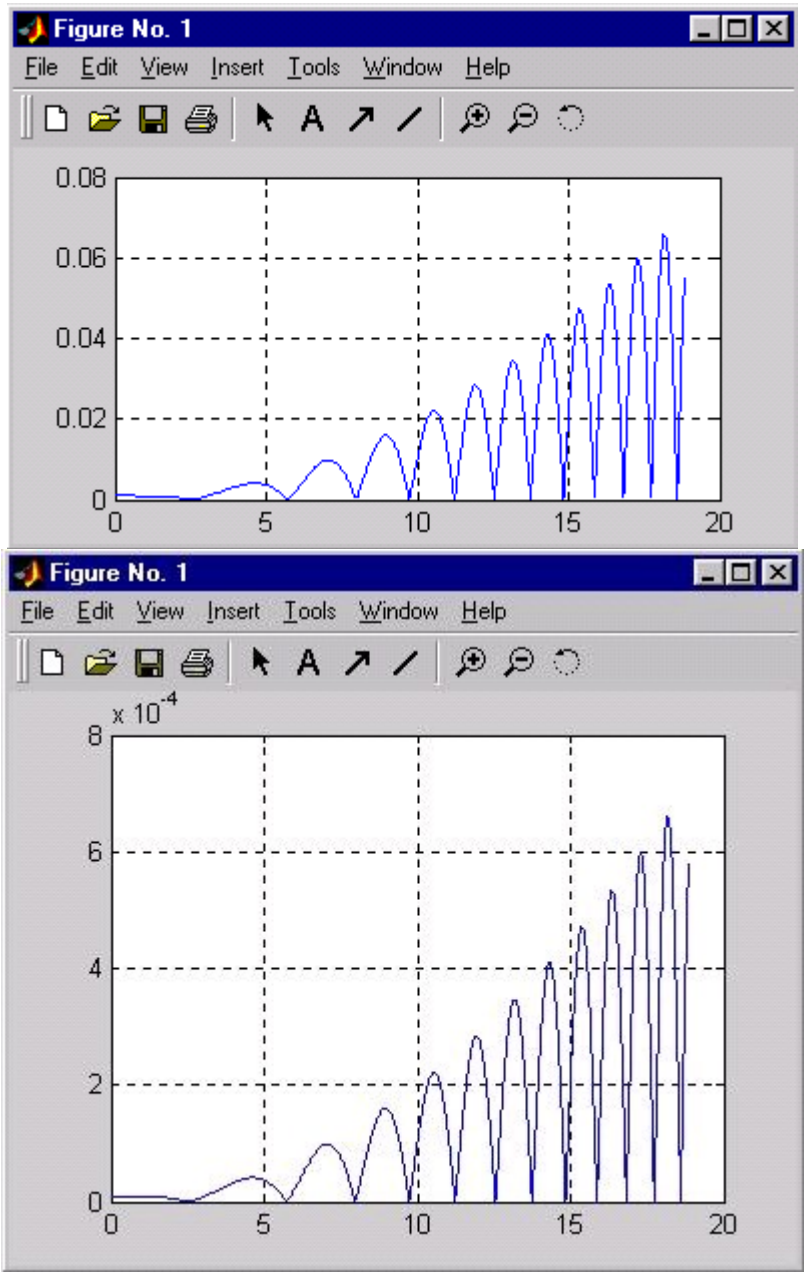
Стек: База

```
1  f=inline ('sin(0.1*x.^2)'); % задание дифференцируемой функции
2  dx=0.01; % шаг координатной сетки
3  x=0:dx:6*pi; % вычисление координат узлов
4  yf=feval(f,x); % вычисление значения функции в узлах
5  N=length(x);
6  m=1:N-1;
7  ddf(m)=(yf(m+1)-yf(m))/dx; % выполнение численного дифференцирования
8  plot(x(m),ddf(m));
9  f1=inline('0.2*x.*cos(0.1*x.^2)'); % задание первой производной
10 ya=feval(f1,x); % вычисление первой производной по аналитической формуле
11 pause
12 plot(x(m),abs(ddf(m)-ya(m))); grid % разность между численным
13                                   % и аналитическим значением производной
```

script Ln 10 Столбец 21

Визуализация производной функции, разность между численными и аналитическими значениями первой производной с шагом сетки $dx = 0.01$ и разность между численными и аналитическими значениями производной с шагом сетки $dx = 0.0001$ представлены на рисунках.





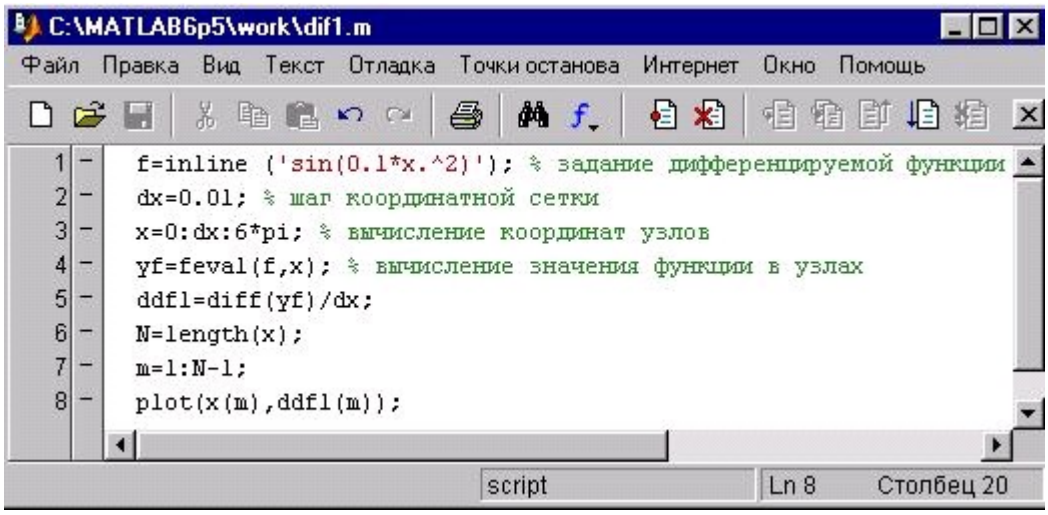
Для аппроксимации производных конечными разностями в пакете Matlab существует функция **diff()**:

diff(x) – возвращает конечные разности, вычисленные по смежным элементам вектора x;

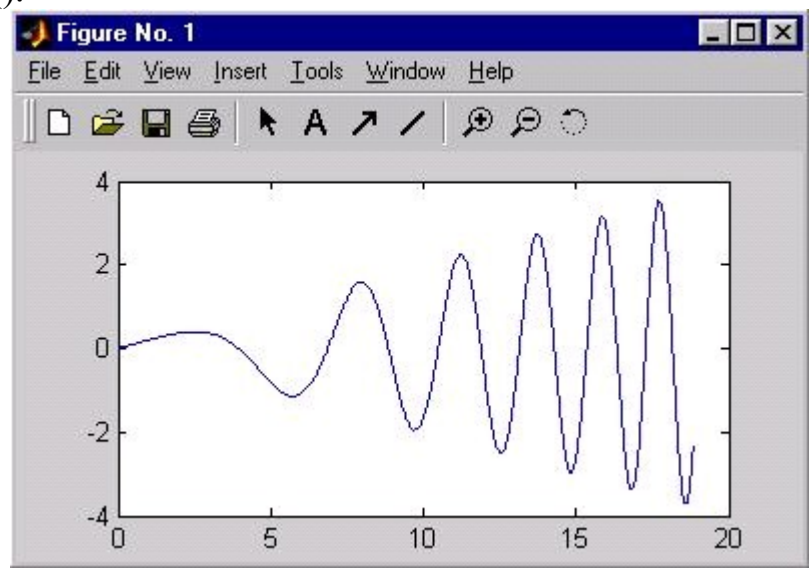
diff(x, n) – возвращает конечные разности n-го порядка, вычисленные по смежным элементам вектора x.

Пример 2. Вычислить значение производной функции $f(x) = \sin(0.1 \cdot x^2)$ на отрезке $[0, 6\pi]$ с использованием соответствующей функции Matlab.

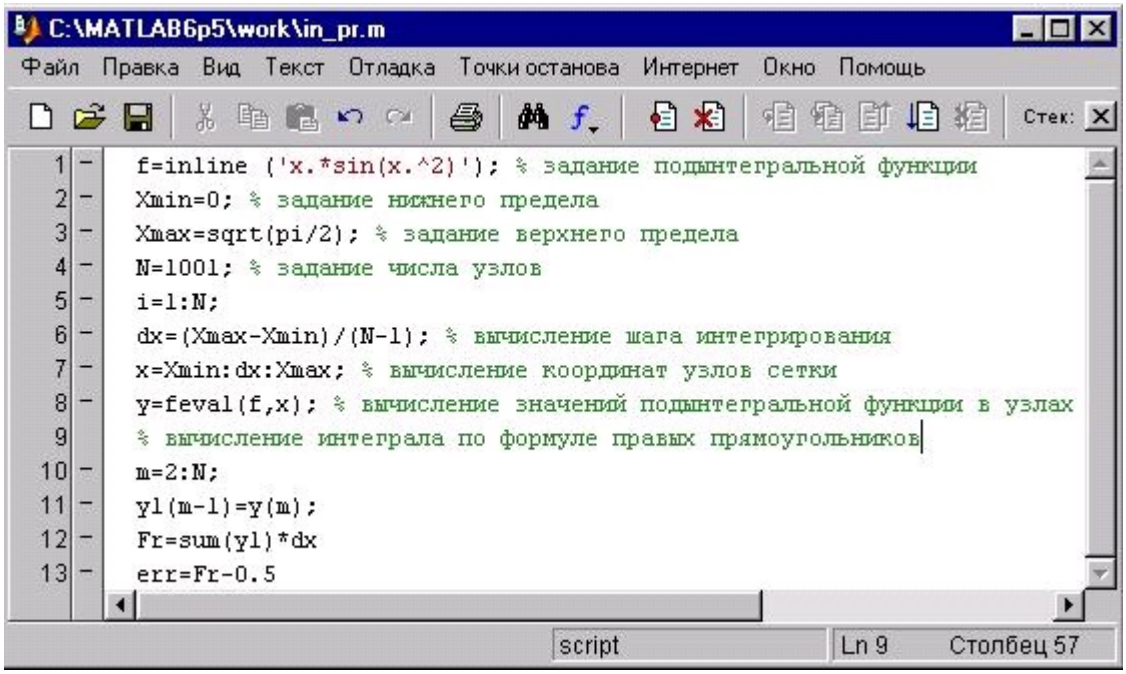
Программа, вычисляющая значение производной с помощью конечных разностей представлена в окне.



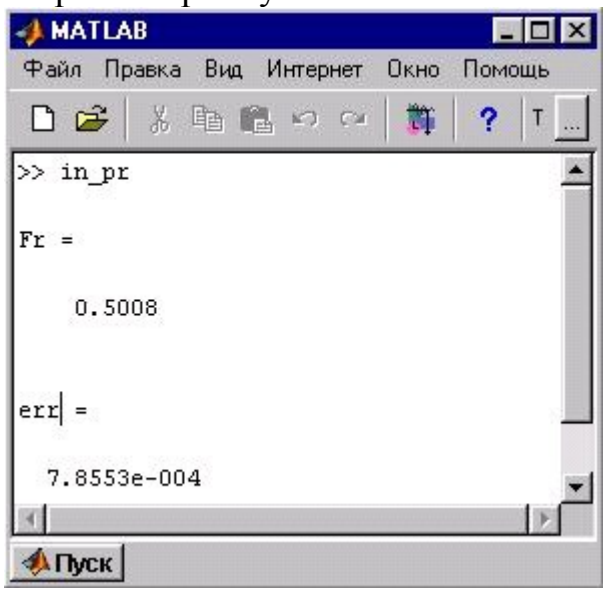
Визуализация первой производной функции, полученной с помощью функции diff().



Пример 3. Вычислить значение определенного интеграла $\int_0^{\sqrt{\frac{\pi}{2}}} x \cdot \sin\left(x^2\right) dx$ методом правых прямоугольников.



В результате работы программы получаем значение интеграла **Fr** и значение ошибки **err** между точным значением интеграла и значением, полученным методом правых прямоугольников.



Пример 4. Вычислить значение определенного интеграла $\int_0^{\sqrt{\frac{\pi}{2}}} x \cdot \sin\left(x^2\right) dx$ методом Симпсона.

C:\MATLAB6p5\work\in_Sim.m

Файл Правка Вид Текст Отладка Точки останова Интернет Окно Помощь

Стек

```
1  f=inline ('x.*sin(x.^2)'); % задание подынтегральной функции
2  Xmin=0; % задание нижнего предела
3  Xmax=sqrt(pi/2); % задание верхнего предела
4  N=1001; % задание числа узлов
5  i=1:N;
6  dx=(Xmax-Xmin)/(N-1); % вычисление шага интегрирования
7  x=Xmin:dx:Xmax; % вычисление координат узлов сетки
8  y=feval(f,x); % вычисление значений подынтегральной функции в узлах
9  % вычисление интеграла по формуле Симпсона
10 s=0;
11 for i=2:N-1
12     if i-2*ceil(i/2)==0
13         k=4;
14     else
15         k=2;
16     end;
17     s=s+k*y(i);
18 end;
19 Fs=(y(1)+s+y(N))*dx/3
20 err=Fs-0.5
```

script Ln 20 Столбец 7

Результат работы программы:

MATLAB

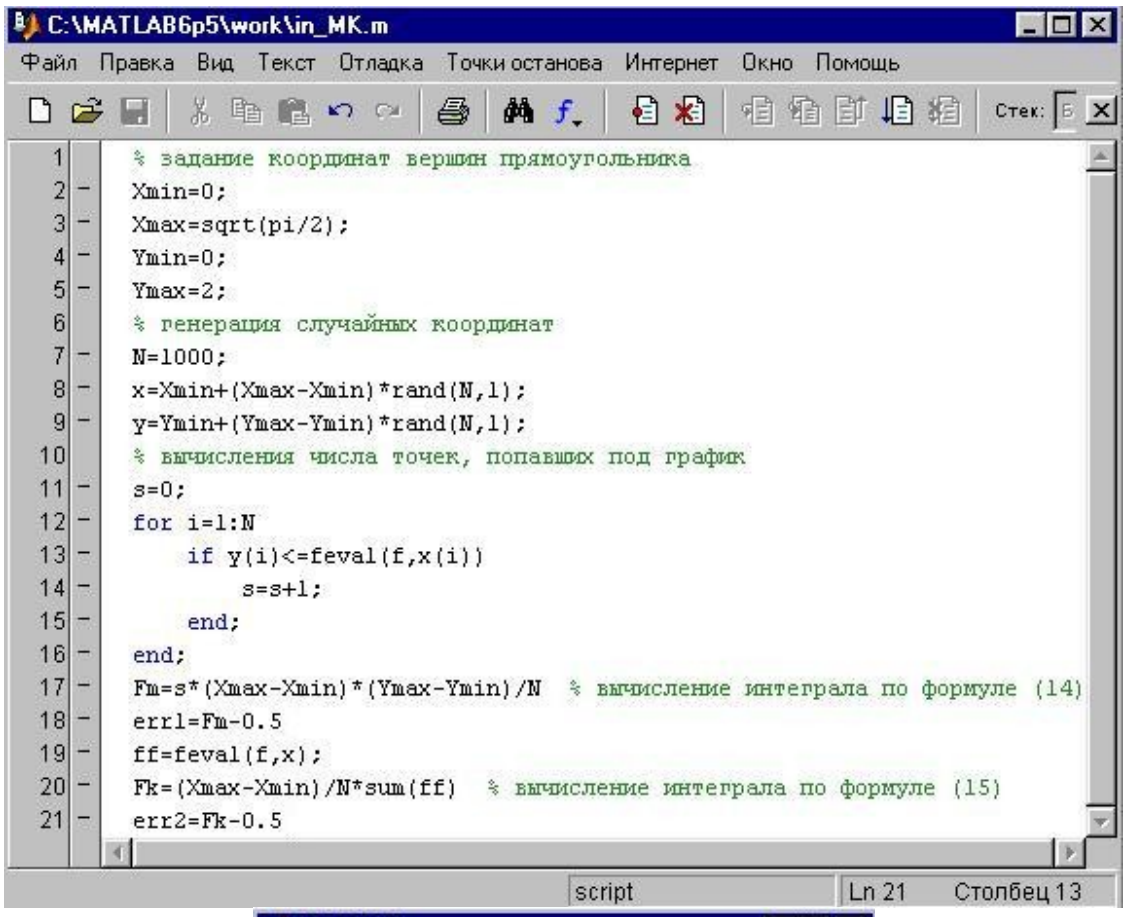
Файл Правка Вид Интернет Окно Помощь

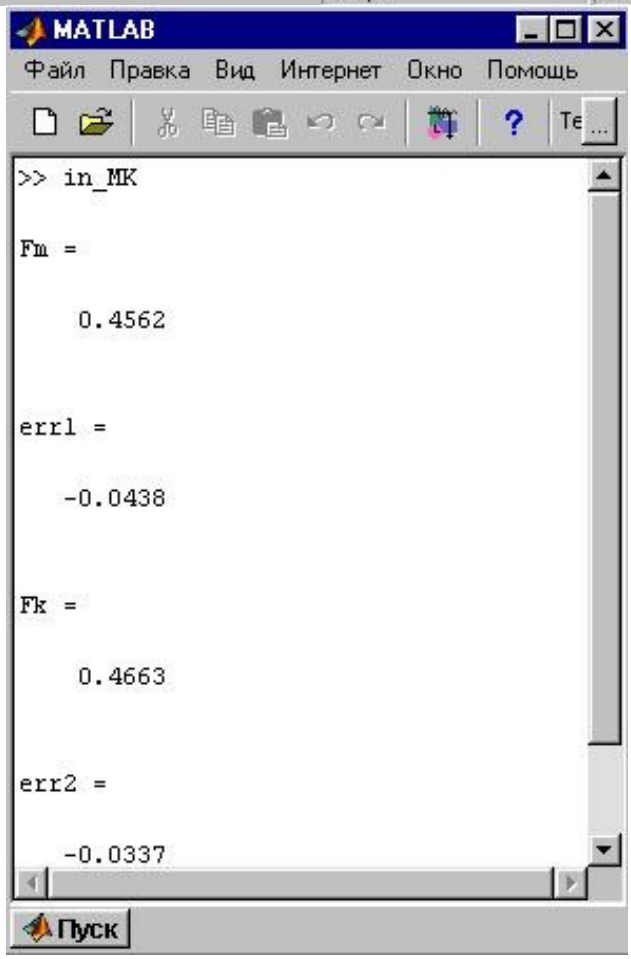
```
>> in_Sim

Fs =

    0.5000


err =

 -5.9908e-013
```

Пуск

Пример 5. Вычислить значение определенного интеграла $\int_0^{\sqrt{\frac{\pi}{2}}} x \cdot \sin\left(x^2\right) dx$ методом Монте-Карло.

В результате получаем значения интеграла и значения ошибок.





Для вычисления значений определенных интегралов в пакете Matlab применяются функции **quad ()**, **quad1 ()**, **trapz ()**.

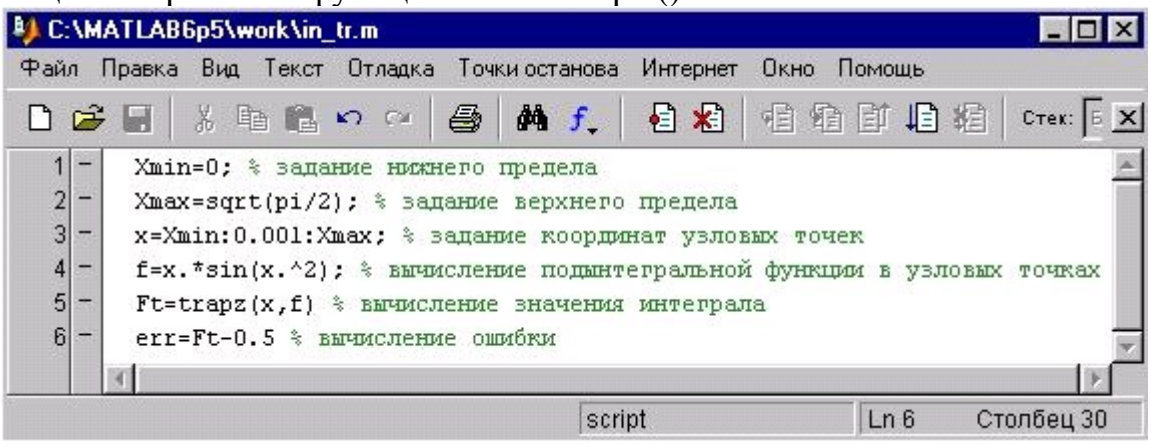
Функция **quad (*fun*, *a*, *b*)** – возвращает значение интеграла от функции *fun* на отрезке $[a,b]$, при вычислении используется метод Симпсона.

Функция **quad1 (*fun*, *a*, *b*)** – возвращает значение интеграла от функции *fun* на отрезке $[a,b]$, используя для вычисления метод Лоббато.

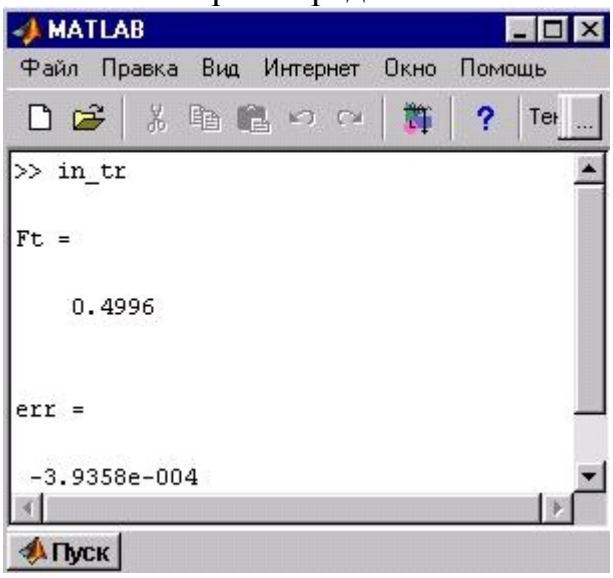
Функция **trapz (*y*)** – возвращает значение определенного интеграла в предположении, что $x = 1 : length(y)$.

Функция **trapz (*x*, *y*)** – возвращает значение определенного интеграла на отрезке $[x(1), x(n)]$.

Пример 6. Вычислить значение определенного интеграла (18) с помощью встроенной функции Matlab trapz ().



Результат вычисления интеграла представлен в окне:



Индивидуальные задания

1. Дана таблица значений функции $y = f(x)$.

x	y	x	y	x	y
1.0	1.2661	1.6	1.7500	2.2	2.6291
1.1	1.3262	1.7	1.8640	2.3	2.8296
1.2	1.3937	1.8	1.9896	2.4	3.0493
1.3	1.4693	1.9	2.1277	2.5	3.2898
1.4	1.5534	2.0	2.2796	2.6	3.5533
1.5	1.6467	2.1	2.4463	2.7	3.8417

2.8	4.1573

Найдите: *а*) значение первой производной функции в точке $x = 0.8 + i \cdot 0.2$, где i – номер вашего варианта; *б*) значение второй производной функции в точке $x = 2.6 - i \cdot 0.1$, где i – номер вашего варианта.

2. С помощью интерполяционных формул Ньютона найдите значение первой и второй производных функции, заданной таблично в пункте 4 вашего варианта, в указанных точках.

Номер варианта	x	Номер варианта	x
1	$x = 1.50,\ x = 1.51,$ $x = 1.59,\ x = 1.60.$	6	$x = 0.50,\ x = 0.51,$ $x = 0.59,\ x = 0.60.$
2	$x = 0.00,\ x = 0.05,$ $x = 0.45,\ x = 0.50.$	7	$x = 1.1,\ x = 1.2,$ $x = 2.0,\ x = 2.1.$
3	$x = 1.0,\ x = 1.1,$ $x = 1.9,\ x = 2.0.$	8	$x = 0.10,\ x = 0.35,$ $x = 2.35,\ x = 2.60.$
4	$x = 0.50,\ x = 0.51,$ $x = 0.59,\ x = 0.60.$	9	$x = 1.50,\ x = 1.51,$ $x = 1.59,\ x = 1.60.$
5	$x = 0.10,\ x = 0.35,$ $x = 2.35,\ x = 2.60.$	10	$x = 0.00,\ x = 0.05,$ $x = 0.45,\ x = 0.50.$

3. Вычислить приближенное значение интеграла по формулам трапеций и трех восьмых, разбивая отрезок интегрирования на 9 одинаковых частей. Вычислить определенный интеграл с помощью функции quad(). Сравнить полученные значения интеграла с точным.

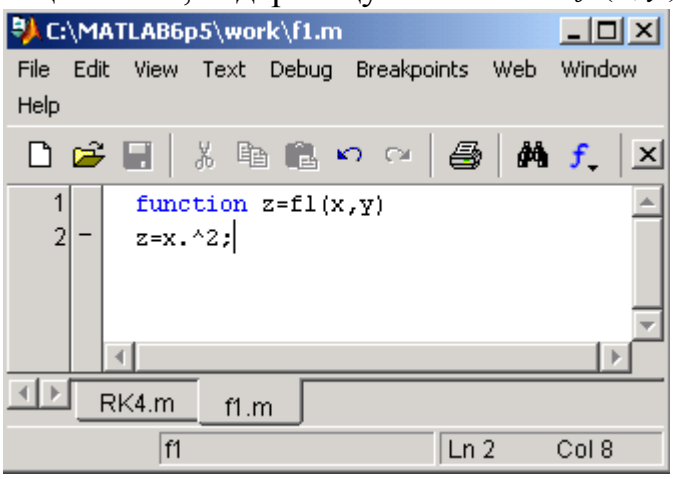
Номер варианта	Интеграл	Номер варианта	Интеграл
1	$\int_1^e \frac{\ln^2(x)}{x} dx$	6	$\int_1^2 \frac{dx}{x^2 + x}$
2	$\int_0^2 \sqrt{4 - x^2}\, dx$	7	$\int_1^{e^{\pi/2}} \cos(\ln(x))$
3	$\int_0^1 \frac{x}{1 + x^4} dx$	8	$\int_1^3 x^3 \sqrt{x^2 - 1} dx$
4	$\int_0^{\frac{\pi}{6}} \frac{\sin^2(x)}{\cos(x)} dx$	9	$\int_{\ln(2)}^{2\ln(2)} \frac{dx}{e^x - 1}$
5	$\int_{-1}^1 x \cdot \arctan(x) dx$	10	$\int_0^1 \frac{2 \cdot \arcsin(x)}{\sqrt{1 - x^2}} dx$

Лабораторная работа № 12.

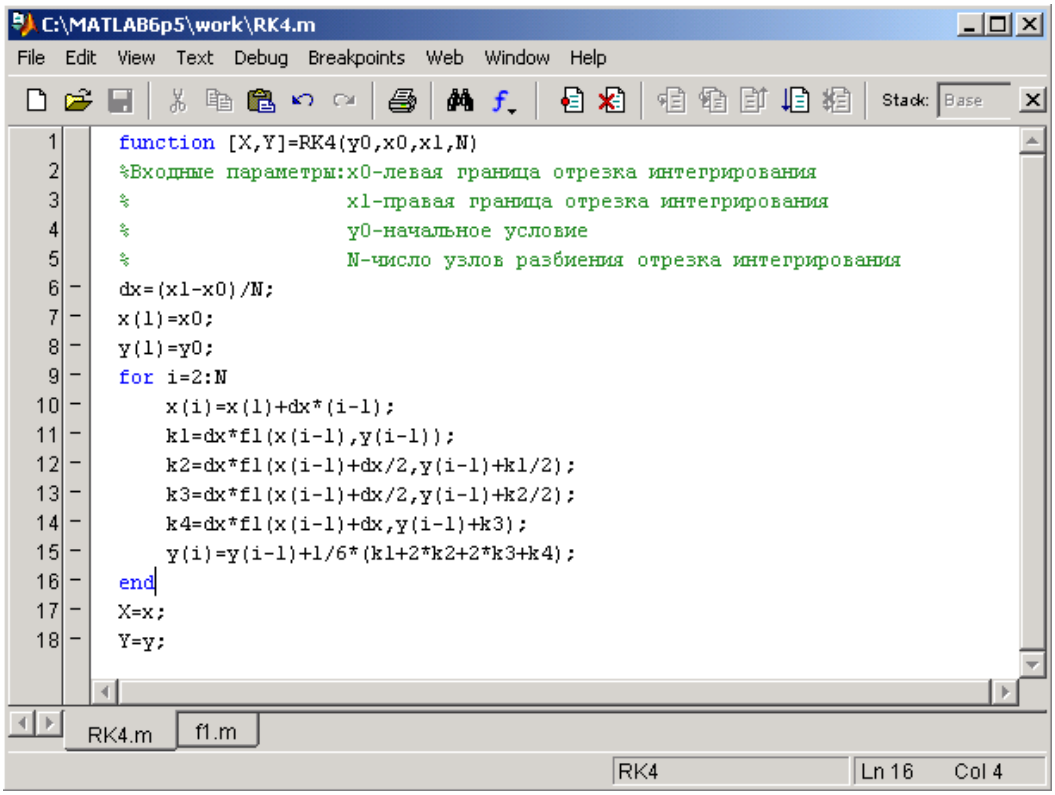
Тема: Численные методы решения обыкновенных дифференциальных уравнений.

Пример 1. Найти решение задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2, \quad y(0) = 1.3$ методом Рунге-Кутты четвертого порядка.

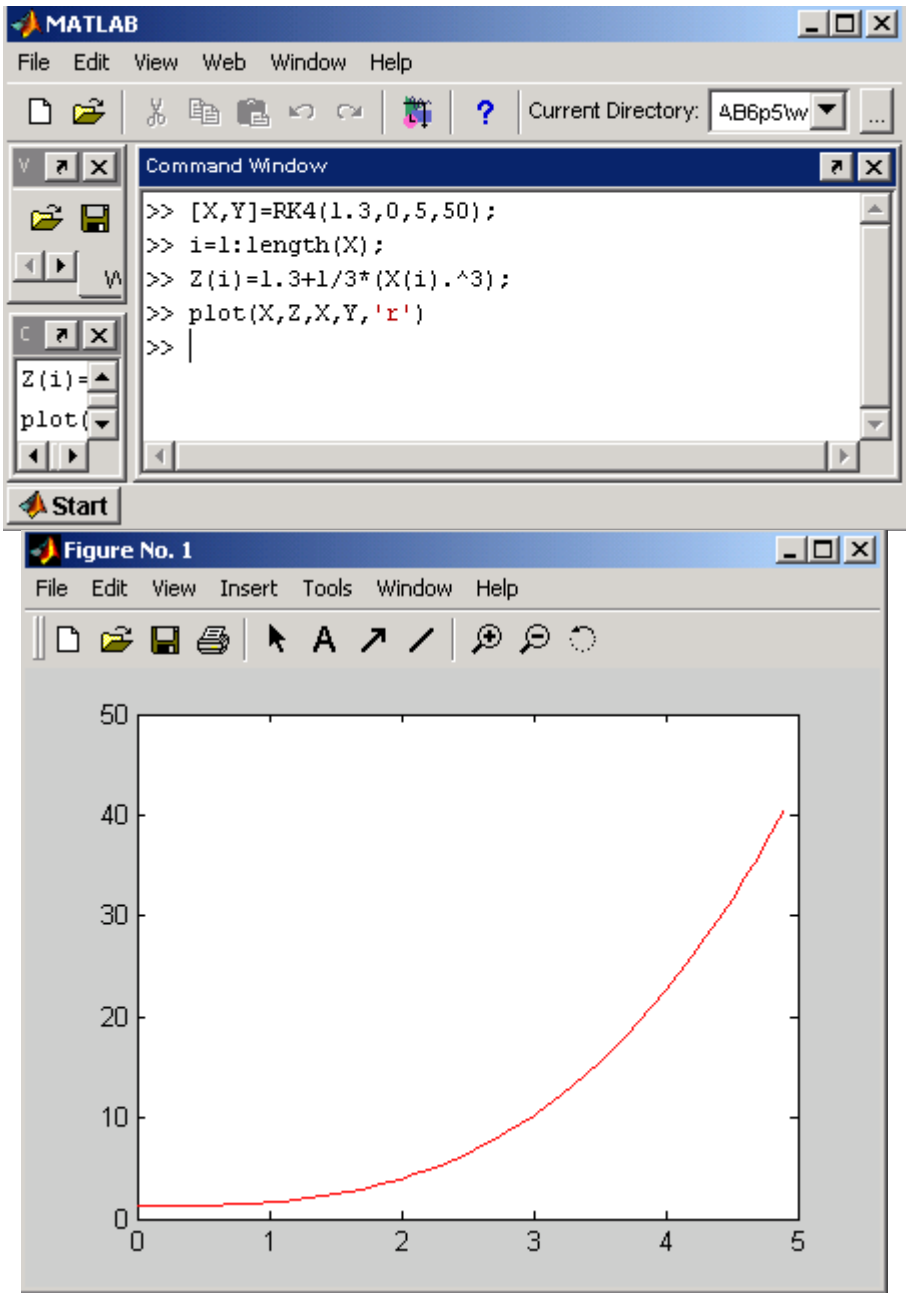
Создадим функцию f1.m, содержащую значение $f(x, y)$



Создадим файл RK4.m, содержащий описание функции, возвращающей решение дифференциального уравнения методом Рунге-Кутты четвертого порядка.



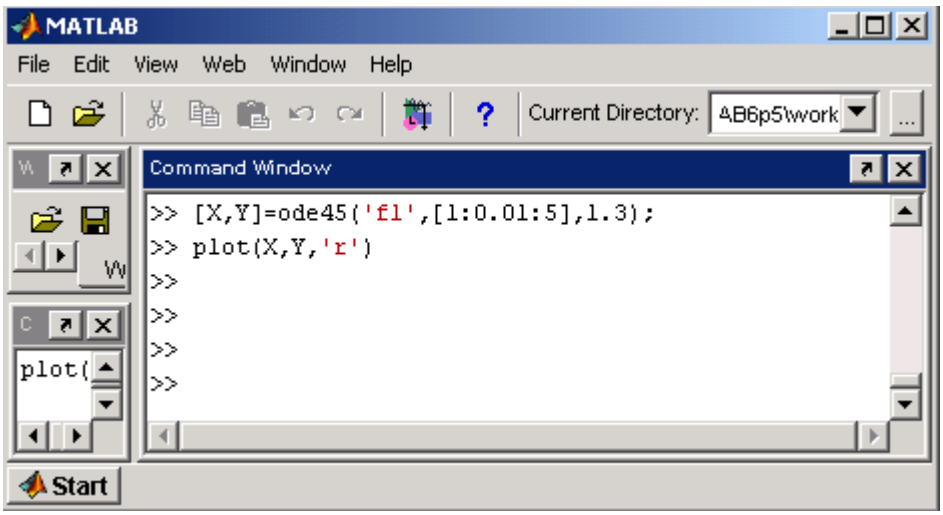
Вызовем функцию RK4.m и построим графики точного и приближенного решений:



Решение задачи Коши для дифференциальных уравнений методом Рунге-Кутты 4 порядка реализовано в пакете Matlab в виде функции **ode 45()**. Описание других функций, реализующих метод Рунге-Кутты и другие методы, а также решатели систем дифференциальных уравнений можно найти, используя справочную систему Matlab.

Пример 2. Найти решение задачи $\frac{dy}{dx} = x^2$, $y(0) = 1.3$, используя встроенные функции пакета Matlab.

Построим решение в Matlab, используя функцию ode45.



Входными данными для функции ode45 являются: имя файла, содержащего определение функции, стоящей в правой части уравнения, вектор, определяющий интервал интегрирования, начальное условие.

Результат построения графика функции $y = f(x)$ аналогичен приведенному выше для точного и приближенного решений.

Индивидуальные задания

1. Применяя указанные методы, реализовать алгоритм численного решения дифференциального уравнения с начальным условием на отрезке $[a,b]$ с произвольно заданным шагом.
2. Решить предложенную задачу Коши, используя встроенные функции пакета Matlab.

Номер варианта	Задача Коши	Отрезок $[a,b]$	Методы
1	$y' = 2x - 3y, \quad y(0) = 1$	[0,10]	Метод Рунге-Кутты 2-го порядка Исправленный метод Эйлера
2	$y' = 0.2y^2 + \frac{0.8}{x^2}, \quad y(1) = 1$	[1,2]	Метод Эйлера-Коши Метод Пикара
3	$y' = \frac{0.2}{x^2} - \frac{y}{x} - 0.8y^2, \quad y(1) = 0.5$	[1,2]	Метод Пикара Метод Рунге-Кутты 4-го порядка
4	$y' = 1.6y^2 + \frac{0.4}{x^2}, \quad y(1) = 1$	[1,2]	Метод Эйлера Метод Рунге-Кутты 4-го порядка

5	$y' = \frac{0.9}{x^2} - \frac{y}{x} - 1.6y^2, \quad y(1) = 0.75$	[1,2]	Метод Эйлера Метод Рунге-Кутты 2-го порядка
6	$y' = 0.9y^2 + \frac{1.6}{x^2}, \quad y(1) = 1$	[1,2]	Исправленный метод Эйлера Метод Пикара
7	$y' = xy^2 + 1, \quad y(0) = 0$	[0,1]	Метод Эйлера Метод Рунге-Кутты 4-го порядка
8	$y' = y^2 + \frac{1}{x^2}, \quad y(1) = 1$	[1,2]	Метод Пикара Метод Эйлера
9	$y' = y^2 + x^2, \quad y(0) = 0$	[0,1]	Исправленный метод Эйлера Метод Рунге-Кутты 4-го порядка
10	$y' = 0.25y^2 + \frac{2}{x^2}, \quad y(1) = 1$	[1,2]	Метод Рунге-Кутты 2-го порядка Метод Пикара

Лабораторная работа № 13.

Тема: Введение в пакет Mathematica.

Начнем знакомство с пакетом **MATHEMATICA** с пары наглядных примеров, хорошо иллюстрирующих возможности применения пакета, и, кроме того, представляющих самостоятельный интерес.

Использованные шрифты несут определенную смысловую нагрузку:

Шрифт "Times New Roman, курсив, разрядка" используется для математических терминов, знание которых необходимо для понимания материала.

"Courier New, жирный" используются для команд пакета MATHEMATICA.

"Times New Roman, курсив" – для важных понятий.

"Times New Roman, курсив, жирный" – для особо важных понятий.

"Times New Roman, жирный, но не курсив" – для обозначения опций пакета MATHEMATICA.

1. Считаем собственные доходы…

Представьте себе, что Вы положили 1 млн. рублей в банк "Золотые горы" под 100% годовых. Для простоты будем считать, что это было 1 января 2003 года. Придя через год, Вы получите в 2 раза большую сумму, – то есть на 1 января 2004 года Вы будете иметь 2 млн. рублей. Если же Вы придете не через год, а через полгода, то есть 1 июля 2003 года, то получите сумму в 1,5 раза большую первоначальной, то есть 1,5 млн. рублей. Еще через полгода она увеличится в 1,5 раза и составит 2,25 млн. рублей. А если приходить в банк каждый месяц? 1 февраля 2001 года сумма Вашего вклада составит 1 млн., увеличенный в $1\frac{1}{12}$ раз, то есть $1\frac{1}{12}$ млн. рублей. Нетрудно видеть, что каждый месяц мы будем увеличивать сумму вклада в $1\frac{1}{12}$ раз. Таким образом, 1 января 2001 года сумма вклада составит $(1\frac{1}{12})^{12}$, то есть, как нетрудно посчитать на калькуляторе, примерно 2 млн. 613 305 рублей. Вы видите, что это значительно больше, чем сумма, полученная в первом или во втором случае. Не ошибка ли это? Ведь при работе с дробями калькулятор производит округления. Проверим наши расчеты в пакете **MATHEMATICA**.

Запустите **MATHEMATICA** версии 3.0 или выше.

Если ее фирменного значка  еще нет на Вашем рабочем столе, найдите файл **Mathematica.exe** с соответствующей пиктограммой и перетащите ее на рабочий стол.

Двойной щелчок по пиктограмме – и пакет запущен. Перед Вами ненадолго появляется заставка, а затем – чистый лист (Notebook, записная книжка) и меню с панелью инструментов наверху.

Просто пишем на нем `(13/12)^12` и нажимаем **Shift+Enter**.

Замечание. В пакете **MATHEMATICA** существуют множество приемов, облегчающих ввод формул. Так, например, для возведения в степень некоторого выражения его нужно выделить и нажать **Ctrl+^**. Скобки рисуются сами собой и курсор переходит в верхний индекс, куда следует ввести показатель степени. **Ctrl+пробел** возвращает курсор в строку.

Вообще, многие клавиши, будучи нажаты вместе с **Ctrl**, создают шаблон формулы – поэкспериментируйте.

Вы увидите следующую картину, изображенную на рис. 13.1.

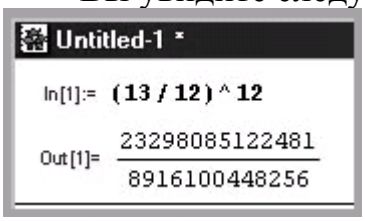


Рисунок 13.1. Фрагмент рабочего документа пакета **MATHEMATICA**.

Перед Вами дробь, которая является абсолютно точным значением суммы Вашего вклада.

Так как Вы задали вопрос в терминах целых чисел, то в них же и выражен ответ. Если бы после числа 13 Вы поставили точку, ответ был бы в 10-ичных дробях: 2.61304, то есть примерно то же, что и было получено с помощью калькулятора. Но мы помним, что это число получено в результате расчетов с округленными значениями.

Как же представить точное значение (дробь) в удобоваримом виде? В пакете **MATHEMATICA** есть возможность записать результаты вычислений со сколь угодно высокой степенью точности (она ограничивается лишь возможностями компьютера). Здесь нужно иметь в виду следующее: в пакете **MATHEMATICA** хранится протокол всех сделанных в процессе данного сеанса работы вычислений (“У меня все ходы записаны”). **In[**номер**]** обозначает ввод, **Out[**номер**]** – вывод при обращении с указанным номером. Так как мы обращались к системе лишь однажды, **Out[1]** содержит полученную нами дробь. Обращение: **N[Out[1]]** будет означать, что результат первого вычисления переводится в десятичную дробь. Точность при этом выбирается автоматически. Обращение: **N[Out[1],10]** будет означать, что мы хотим получить в результате 10 знаков (всего). Такой же результат был получен нами с помощью калькулятора. Как и при работе с калькулятором, мы можем использовать результаты предыдущего вычисления. Обращение к предыдущей выходной ячейке осуществляется с помощью знака процента: **%**. Так, ввод **N[%,10]** на втором шаге эквивалентен **N[Out[1],10]**.

Замечание. Чтобы заставить несколько команд восприниматься вместе (и тем самым избежать вывода результатов выполнения (**Out**’ов) каждой из них), нужно разделять их символом “**;**”.

Вернемся в наш банк “Золотые горы”. Что если пересчитывать вклад каждый день, а еще лучше – каждый час, каждую минуту, каждую секунду? Создается впечатление, что если пересчет будет производиться очень часто, то через год сумма возрастет в очень много раз. Хватит ли принесенных нами мешков, чтобы унести все деньги? Прикинем: при ежесекундном пересчете вклада итоговая сумма составит 2 млн. 718 281 рубль. Хм, мы думали, будет больше. Возникает смутное подозрение – не существует ли предела этой суммы при сколь угодно частом посещении банка? Разберемся: пусть n – количество наших временных интервалов в году (дней, минут, секунд…). Соответствующие формулы имеют вид: $(1+\frac{1}{n})^n$. Существует ли $\lim(1+\frac{1}{n})^n$ при $n\to\infty$? Спросим об этом пакет **MATHEMATICA**. Сначала выясним, знает ли

MATHEMATICA пределы. Это можно сделать с помощью электронного учебника, встроенного в пакет и вызываемого при помощи опции **Help**.

Нужно сказать, что **MATHEMATICA** – очень дружественный пакет. В нем существует несколько способов поиска вспомогательной информации. Так, достаточно набрать **`??Limit`**, (или, например, только **`??Li*`** или **`??*imit`**), и перед нами появится список всех ключевых слов, содержащих данную подстроку. Выделим мышкой нужное нам слово и нажмем **F1** – откроется соответствующая страница **Help**. Можно было сделать и так: **Help** – **Help** – ввести слово **Limit** в предлагаемое поле и нажать **Enter**.

Help содержит не только описание синтаксиса обращения к функции, но и, что очень удобно, примеры ее использования. Так, мы увидим пример вычисления предела, общий вид обращения к функции вычисления предела: **`Limit[expr,`** $x \to x_0$ **`]`**, описание действия оператора и (в рубрике **Futher Examples**) соответствующие примеры.

Замечание. В пакете **MATHEMATICA** стремление аргумента обозначается знаком “- > ” (дефис и “строго больше”), бесконечность – словом **Infinity**. Стрелочку и бесконечность можно изобразить покрасивее, если до и после “ - > ” и “**inf**” нажать **Esc**. Если одни и те же операции вызываются неоднократно, удобно использовать входящие панели инструментов пакета (**Palettes**). Их вызов осуществляется через меню: **Files** – **Palettes**. (см. рис. 13.2.) В палитре 2 **Basic Calculations** – **Calculus** – **Common Operations** существует шаблон для задания предела, в 5 **Complete Characters** – знаки бесконечности и стремления.

Очень важное замечание. Для ввода с помощью шаблона лучше предварительно задать стиль входной ячейки как **Standard**: **Cell – Convert To – StandartForm.**

На рисунке 13.2 показан результат применения двух способов написания выражения для вычисления предела: *ручного*, требующего по словам известного физика Р. Йоста, “рудиментарного владения латинским и греческим алфавитами”, и *шаблонного*. Там же изображены две инструментальные палитры – отыщите на них шаблон для показателя степени и символ бесконечности.

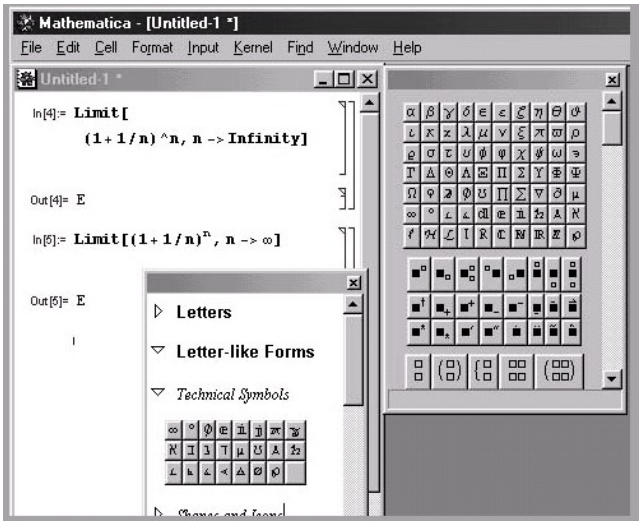


Рисунок 13.2. Фрагмент рабочего документа и две палитры.

Итак, сколь бы часто мы ни пересчитывали сумму вклада, существует предел ее увеличения. Он равен числу **e**, свойства которого были установлены в 18 веке великим математиком Леонардом Эйлером.

Здесь речь шла о 100% годовых. К сожалению, такая ставка нереальна даже для банка “Золотые горы”. Предположим, что процентная ставка равна *x*. (Ранее *x*=1.) Будет ли существовать предел суммы вклада в этом случае? Опять обратимся к пакету **MATHEMATICA**. Результатом вычисления: **Limit[(1+x/n)^n,n->Infinity]** является e^x. Таким образом, независимо от процентной ставки сумма вклада будет стабилизироваться при сколь угодно большом значении *n*. При каждом значении ***x*** мы будем получать свое значение предела. Это и будет та предельная сумма, которую Вы получили бы через год после оформления вклада, если бы вообще не покидали стен банка и непрерывно осуществляли пересчет вклада.

Что же делать тем вкладчикам, у кого нет под рукой компьютера с пакетом **MATHEMATICA** и даже нет калькулятора? Давайте сделаем для них следующее: распечатаем в пакете **MATHEMATICA** график предельной суммы как функции от процентной ставки ***x***. С этим графиком они для любого значения ***x*** легко узнают предел своих финансовых возможностей. Построение графиков (функция **Plot**) – одна из наиболее используемых возможностей пакета **MATHEMATICA**. Набираем: **Plot[E^x,{x, 0,1.2}]** и получаем график, изображенный на рисунке 13.3.

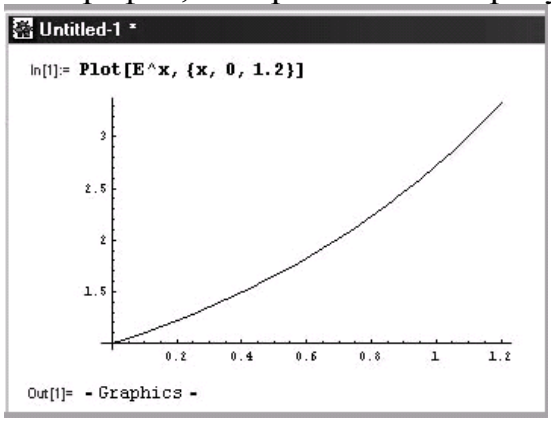


Рисунок 13.3. График зависимости предельного годового дохода (т.е. дохода, полученного в конце года при бесконечно частом пересчете суммы вклада) от процентной ставки.

Процентная ставка изменяется здесь от 0 до 120%.

График хорош, но можно сделать его еще более понятным и легко читаемым. Для этого существует множество параметров функции **Plot**. Им

можно присваивать значения с помощью значка “**->**” (rule, правило): **Plot[**присвоения через запятую**]**. В таблице 13.1 описаны некоторых из параметров функции **Plot**.

Таблица 13.1. Параметры функции **Plot**.

PlotLabel->"INCOME DEPENDING ON X RATE"
Заголовок графика
PlotRange->{0,3.5}
Диапазон изменения переменной y на графике. (По умолчанию он выбирается так, чтобы показать характер поведения функции на всем заданном интервале изменения аргумента.) Если точка *y*=0 не попадает в этот диапазон, ось абсцисс переносится в нижнюю или верхнюю (в зависимости от функции) его границу.
AxesLabel->{"x","E^x"}
Надписи на осях абсцисс и ординат.
Ticks->{{0.0,0.2,0.4,0.6,0.8,1.0,1.2,1.4}, {0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5}}
Разметка осей абсцисс (первый список) и ординат (второй список). Точки, не попадающие в диапазон изменения переменных *x*, *y*, не отображаются.
GridLines ->{0.0,0.4,0.8,1.2},{0.0,0.5,1.0,1.5,2.0,2.5}}
Линии вспомогательной решетки – абсциссы ее узлов заданы компонентами первого списка, ординаты – второго. Отображаются лишь те участки линий решетки, которые попадают в диапазон изменений переменных *x*, *y*. Заметим, что значения свойств **Ticks** и **GridLines** не связаны друг с другом.
PlotStyle->{Thickness[0.009],RGBColor[1,0,0]}
Стиль линии графика. Здесь заданы параметры: толщина линии как 0.009 ширины рисунка и цвет графика (красный).
AxesStyle->{Thickness[0.005],RGBColor[0,1,0]}
Стиль осей. Здесь задан один только список параметров, он используется для обеих осей. Можно было задать разные стили для оси абсцисс и оси ординат, (создав для этого два списка параметров), но мы поленились.

Постройте самостоятельно окончательный вид нашего графика.

Сохраним записную книжку с помощью опции **Save** меню **File**. При этом создается файл с расширением **.nb**. Пакет **MATHEMATICA** позволяет работать одновременно с несколькими файлами (записными книжками). При этом нумерация всех “ходов” является сквозной. Если Вы закрыли все записные книжки, то открыть новую записную книжку можно опцией: **File** – **New**.

Теперь Вы можете получить протокол всего Вашего диалога с пакетом. Однако, вся “портянка”, как правило, бывает не нужна. Нужно убрать из диалогового окна лишние фрагменты, написать заголовок, комментарии, привести соответствующие формулы. Обратите внимание: справа есть вертикальные линии, выделяющие на экране отдельные ячейки и их группы. Вы можете делать с ячейками все, что хотите – удалять их, свертывать,

объединять, управлять их стилем – тем самым содержимое экрана может быть оформлено так, как Вы того пожелаете. Пока мы имели дело с ячейками типа Input, Output, Standard. Существуют и другие типы: заголовки (Title, Subtitle,...), текст, формулы и пр. Они выполняют функции соответствующих опций текстовых редакторов, позволяя красиво оформить содержимое записной книжки. Основные возможности работы с ячейками в пакете **MATHEMATICA** перечислены в таблице 13.2.

Таблица 13.2. Работа с ячейками

Что сделать с ячейкой	**Как это сделать**
Выделить	Подвести курсор (он изменит свою форму на тонкую стрелку с вертикальной чертой впереди) к скобке ячейки и щелкнуть мышью.
Удалить	Нажать **Delete**. Выделенная ячейка будет удалена.
Сгруппировать	Существует два режима группировки: автоматический и ручной. Выбор режима осуществляется опцией **Cell – Cell Grouping**. При автоматическом режиме (**Automatic Grouping**) группируются входная и выходная ячейки. При ручном режиме (**Manual Grouping**) можно сгруппировать любое количество смежных ячеек. Для этого их следует выделить и выбрать опцию **Cell – Cell Grouping – Group Cells**.
Свернуть/Развернуть (для группы ячеек)	Двойной щелчок на соответствующей скобке.
Задать стиль ячейки	**Format – Style** – соответствующая опция.

Итак, мы рассмотрели один пример применения пакета **MATHEMATICA**. Отметим некоторые свойства пакета, на которые Вы, должно быть, уже обратили внимание:

а) Все выражения (имена функций, параметров, обозначения известных констант) пишутся с большой буквы. Например: **E**, **Infinity** (бесконечность), **Pi**, **GoldenRatio** (отношение золотого сечения: кстати, именно таким является принятое по умолчанию соотношение длины и ширины прямоугольной области, на которой размещается график).

б) Все аргументы функций заключены в квадратные скобки. Строки заключены в кавычки. Некоторые параметры функций определяются *совокупностью свойств*.

в) Ввести одну и ту же формулу можно разными способами: набором слов с клавиатуры или с помощью шаблона соответствующей панели.

Мы использовали лишь малую часть возможностей пакета **MATHEMATICA** – для анализа нашего простого примера потребовались лишь самые основные его средства. Далее мы остановимся еще на некоторых возможностях пакета, широко используемых при проведении различных вычислений, их визуализации, оформлении результатов.

Лабораторная работа № 14.

Тема: Элементарная статистика в пакете Mathematica.

2. Считаем чужие доходы

Вы слышите по местному радио, как мэр Вашего городка утверждает, что средний доход его жителей составляет 1 000 у.е. Вы возмущены! Самым “популярным” среди Ваших многочисленных родственников, знакомых, знакомых ваших родственников и родственников Ваших знакомых является доход в 500 у.е. Хотя, чуть поразмыслив, Вы понимаете, что такое возможно. Если олигархи (т.е. 1% населения) получают, скажем, по 100 000 у.е., то даже при нулевом доходе остального населения (99%), средний доход будет составлять: $0.01 \cdot 100\,000 = 1\,000$ (у.е.). Говоря языком математической статистики, наиболее распространенное значение дохода – *мода*, – не совпадает со средним его значением, или математическим ожиданием.

Через некоторое время Вы получаете повышение по службе, и Ваш доход вырастает до 800 у.е. Вы замечаете, что половина Ваших сослуживцев Вам завидует, а половина смотрит свысока. Значит, Вы – типичный представитель среднего класса! Но доход-то у Вас гораздо ниже среднего! Это Вам не дает покоя, и Вы достаете учебники, где излагаются основные сведения из теории вероятностей и математической статистике. Теперь мы уверены в том, что Вы владеете основными понятиями из теории вероятностей (непрерывные и дискретные *случайные величины* (с.в.), *плотность* и *функция распределения*, числовые характеристики с.в.: *среднее*, *дисперсия, среднее квадратическое отклонение*, *квантили*), а также из математической статистики (*выборка*, *генеральная совокупность*, *выборочные оценки* параметров распределения).

Итак, 800 у.е. – это *медиана* распределения дохода, или 50% *квантиль*. Перед нами три (!) различные характеристики с.в., причем все они в некотором смысле характеризуют средний ее уровень. Это все понятно. Но среднее значение дохода в обществе – это все равно, что средняя температура пациентов в больнице. Чтобы оценить состояние общества, нужно знать закон распределения дохода, а не только его отдельные параметры.

Опишем закон распределения дохода для Вашего города и проанализируем его взаимосвязь с модой, медианой и математическим ожиданием.

Одним из наиболее распространенных распределений вероятностей является так называемое *нормальное*, или *Гауссово* распределение. Плотность нормального распределения с параметрами (***a, σ***) имеет вид:

$$p_{\eta}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}.$$

По нормальному закону распределены многие с.в., совершенно разные по своей физической сути. Не подойдет ли оно для описания распределения доходов?

Как известно, мат. ожидание, мода и медиана нормального распределения совпадают с параметром ***a***. (Вы можете легко убедиться в

этом, используя очевидное равенство $p_\eta(a+t) = p_\eta(a-t)$.) Для нашего же случая мат. ожидание, мода и медиана дохода различны. Значит, доход в нашем городе явно распределен ненормально! Не будем призывать к перераспределению доходов, а займемся лучше исследованием свойств других распределений.

Как известно, нормальным является распределение суммы большого числа независимых факторов, – в этом случае говорят об *аддитивном* характере их воздействия на результирующую с.в. В экономике же характер совместного действия многочисленных случайных факторов нередко является не аддитивным, а *мультипликативным*: значение результирующего признака η, достигнутое за счет действия случайного фактора ξ, пропорционально текущему значению η с коэффициентом $(1+\xi)$. (Так, в примере из раздела 1, сумма вклада в банке "Золотые горы" на 1 февраля 2003 года пропорциональна сумме вклада на 1 января 2003 года с коэффициентом $(1+\xi)$, где ξ равна 1/12 ставки процента на 2003 год.) Если число случайных факторов достаточно велико, воздействие каждого из них незначительно и имеет мультипликативный характер, то результат будет иметь не нормальное, а так называемое логарифмически-нормальное (логнормальное) распределение.

Плотность логнормального распределения имеет вид:

$$p_\eta(x) = \frac{1}{\sigma x\sqrt{2\pi}} e^{-\frac{(\ln x - \ln a)^2}{2\sigma^2}} \qquad (14.1)$$

Функция распределения:

$$F_\eta(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^{\ln x} e^{-\frac{(t-\ln a)^2}{2\sigma^2}} dt \qquad (14.2)$$

Упражнение 14.1. Докажите, что логарифм с.в., распределенной по логнормальному закону с параметрами (***a, σ***), имеет нормальное распределение с параметрами (***ln a, σ***). Кроме того, покажите с помощью пакета **MATHEMATICA**, что плотность логнормального распределения есть производная функции (14.2) и начертите ее график при значениях параметров ***a=800*** и ***σ=0.685568*** и сравните его с графиком на рисунке 14.2.

Указание (причем очень важное): MATHEMATICA – высоко интеллектуальный пакет, умеющий дифференцировать и интегрировать аналитически заданные функции. Соответствующие команды:

D[дифференцируемая функция, аргумент**]**,

Integrate[подынтегральная функция, {аргумент, нижний предел, верхний предел}**]**.

Замечание. Если Вам лень набирать команды на клавиатуре, используйте шаблоны из панели: **2 Basic Calculations** (см. рис. 13.2). Вписывать формулы в квадратики шаблона – работа почти ювелирная. Облегчить ее можно за счет "внедрения" подынтегральной функции в placeholder. Для этого нужно выделить формулу и нажать шаблон интеграла. Перемещение курсора между нижним и верхним пределами интегрирования осуществляется с помощью клавиш: **Ctrl+%** (**Ctrl+5**).

Другие характеристики логнормального распределения:

Среднее значение (математическое ожидание):

$$M(\eta) = ae^{\frac{\sigma^2}{2}}$$

Мода: (14.3)

$$Mode(\eta) = ae^{-\sigma^2}$$

Медиана:

$$Median(\eta) = a$$

Из перечисленных выше формул видно, что куб медианы логнормального распределения равен произведению квадрата среднего на моду. В нашем примере это равенство выполняется приблизительно: $800^3 \approx 1000^2 \cdot 500$.

Проведем статистический анализ распределения дохода

В пакете **MATHEMATICA** есть отдельные модули (packages) для выполнения различных специальных вычислений. Чтобы выполнить статистические расчеты, подключим "мастера", который будет сам осуществлять поиск нужного модуля: **<<Statistics`Master`**. (Обратите внимание, что здесь используются обратные апострофы – они расположены на клавиатуре под знаком "тильда".)

Замечание. Чтобы избежать синтаксических ошибок при вводе названий законов распределений, лучше использовать **Help** и копировать оттуда целые выражения. Удобно также не набирать длинные ключевые слова целиком – после нескольких введенных символов нажмите **Ctrl+K** и выберете нужный пункт из возникающего при этом списка. Однако это не срабатывает для ключевых слов отдельных модулей.

Подготовим "почву" для последующего анализа распределения дохода. Так как мы будем неоднократно обращаться к логнормальному закону, его удобно было бы как-то запомнить. Пакет **MATHEMATICA** предоставляет нам такую возможность – переменными в нем могут быть не только числа или строки, но и другие объекты. Найдем нужное нам распределение в общем списке понятных **MATHEMATICe** законов: **Help – Help – Add-ons – Standard packages – Statistics – ContiniousDistributions: LogNormalDistribution[mu,sigma]**. Найдем значения параметров. Будем считать, что значения параметров совпадают с их *точечными оценками*. Для нашего примера число 800 является оценкой для медианы, то есть параметра ***a*** в формулах (14.3). Величина ***lna*** является значением параметра **mu** логнормального распределения в обозначениях пакета **MATHEMATICA** (а вовсе не значением математического ожидания логнормально

распределенной с.в., как написано в **Help**'е 3-ей версии пакета **MATHEMATICA** – в 4-ой версии эта ошибка уже исправлена). Значение параметра **sigma** найдем из условия: $\frac{Median(\eta)}{Mode(\eta)} = e^{\sigma^2}$. Получаем: **sigma**=0,685568. Запомним выбранный закон с указанными параметрами как значение некоторой переменной (назовем ее **mylog**):

```
mylog=LogNormalDistribution[Log[800], 0.685568]
```

Замечание. Имя переменной в пакете **MATHEMATICA** может быть любой последовательностью латинских строчных или прописных букв, цифр и знака $. Строчные и прописные буквы различаются.

Теперь мы можем ответить на многие вопросы. Например, нас интересует, *какой процент населения имеет доход выше среднего*. Иначе говоря, нас интересует *вероятность* превышения случайной величиной значения ***x***, равного среднему доходу. Из определения функции распределения с.в. ($F_\eta(x)$) эта вероятность есть $1 - F_\eta(x)$. Так как средний доход равен 1 000 у.е., запишем: **1-CDF[mylog,1000]**. Здесь **CDF[distribution,x]** – значение функции выбранного распределения (**distribution**) в точке **x**.) Получим: **0.372406**. Следовательно, доход выше среднего имеют 37,24% жителей города.

А сколько человек имеют очень маленький доход, скажем, не превышающий 400 у.е.? **CDF[mylog,400]**: почти 16%.

Какой процент населения имеет доход свыше 10 000? Получается, лишь чуть больше 0,1%. Это противоречит Вашим представлениям об олигархах – в него, по-вашему, входит каждый сотый житель, и половина из них имеет доход свыше 10 000.

Какой же доход является "проходным баллом" к олигархам? Эта задача обратна предыдущей. Мы ищем такое значение дохода ***x***, что вероятность иметь меньший доход составляет 99%. Иначе говоря, мы ищем корень уравнения $F_\eta(x) = 0,99$, или 99% *квантиль*:

Quantile[mylog,0.99] дает 3 942 "с хвостиком". Вам же кажется, что "проходной балл" должен быть существенно выше.

Попробуем разобраться. Мы видим, что распределение дохода в Вашем городке, возможно, подчинено логнормальному закону. По крайней мере, если рассматривать лишь ту категорию населения, доход которой расположен *левее медианы* (условимся называть их ***бедными***), то все полученные результаты неплохо согласуются с Вашими представлениями. Что же касается людей, чей доход *превышает медиану* (их будем называть ***богатыми***), то здесь дело обстоит несколько иначе, чем Вам представляется. Богатым, похоже, логнормальный "закон не писан". Отметим, что согласно нашему определению бедные (как и богатые) составляют половину населения.

Считать чужие деньги – занятие интересное не только для нас с Вами. Еще в конце прошлого века итальянский экономист Вильфредо Парето рассматривал распределение дохода налогоплательщиков. Поскольку часть

населения налогов не платит (ввиду того что их доход не превышает установленного значения), Парето имел дело лишь с теми значениями, которые превышают этот порог. (Позднее такие распределения были названы *усеченными*.) Парето обнаружил, что закон распределения высоких доходов описывается формулой:

$$p_{\eta}^{(Парето)}(x) = \frac{\alpha}{c}\left(\frac{c}{x}\right)^{\alpha+1}; \quad x \geq c\,. \tag{14.4}$$

Здесь параметр c обозначает минимальное значение с.в., а параметр α характеризует кривизну функции.

Итак, мы хотим описать распределение дохода среди богатого населения. Значение параметра c плотности распределения Парето, таким образом, равно медиане (800 у.е.), а значение параметра α нам нужно оценить. Получить необходимые для этого промежуточные результаты Вы можете, сделав следующие упражнения:

Упражнение 14.2. Найдите вручную формулы мат. ожидания и функции распределения Парето и проверьте результат с помощью пакета **MATHEMATICA**.

Указание: для нахождения математического ожидания удобно использовать функцию **Mean** статистического модуля пакета **MATHEMATICA**.

Упражнение 14.3. Для данных нашего примера убедитесь с помощью пакета **MATHEMATICA** в том, что средний доход бедной части населения примерно равен моде.

Указание: плотность логнормального распределения нужно умножить на 2х (т.к. бедное население составляет лишь половину общей численности) и взять от нее интеграл в пределах: 0 – 800.

Оценим значение параметра α, исходя из среднего дохода богатой части населения. Поскольку бедная часть населения имеет средний доход, примерно равный 500 у.е. (см. упражнение 14.3), общий средний доход равен 1 000 у.е., численность богатого и бедного населения одинакова, то средний доход богатых равен 1 500 у.е. Формула для мат. ожидания распределения Парето (см. упражнение 14.2) имеет вид:

$$M(\eta) = \frac{\alpha}{\alpha - 1}c, \ \alpha > 1 \tag{14.5}$$

Отсюда получаем: $\frac{\alpha}{\alpha-1}\cdot 800 = 1\,500 \Rightarrow \alpha = 2.1486$.

Запомним закон Парето с нашими параметрами:

```
myPareto=ParetoDistribution[800,2.1486].
```

Попытаемся произвести “стыковку” законов распределения дохода бедного и богатого населения. Напомним, что по нашей классификации бедных и богатых поровну, а потому графики плотностей распределения дохода должны совпасть в медиане.

Важное замечание. Как известно, плотность *любого* распределения ограничивает на плоскости фигуру *единичной* площади. Мы используем “половину” логнормальной кривой для описания распределения доходов бедного населения. Плотность же распределения Парето перед “стыковкой” необходимо уполовинить, то есть домножить его на долю богатого

населения. Тогда результирующая кривая будет ограничивать фигуру площадью ½+½=1.

Вычислим значение плотности распределения Парето в медиане (x=800): **PDF[myPareto,800]/2**. Получаем число 0.00134288. Вычислим при x=800 значение плотности логнормального распределения:
PDF[mylog,800] есть 0.000727394, что существенно меньше значения плотности распределения Парето. Таким образом, ожидаемой "стыковки" не происходит.

Чем же это объяснить? Напомним, что у нас нет реальных данных, и все выводы о параметрах распределения дохода мы делаем на основании: а) известного *математического ожидания* (мы не смеем подвергать сомнению правильность утверждений мэра); б) приблизительного значения *моды* (полученного по Вашим оценкам); в) еще гораздо более приблизительной оценки *медианы* (основанной лишь на эмоциональной окраске Ваших контактов с сослуживцами). Похоже, Вы не совсем точны в своих оценках – медиана, видимо, расположена левее, (то есть по нашей двоичной классификации Вы уже относитесь к категории богатых).

Уточним медиану распределения дохода

Напомним, что бедняки подчиняются логнормальному закону, а богачи живут по закону Парето. В качестве исходных данных будем использовать *моду* и *математическое ожидание*. Из формул (14.3) получим выражение для медианы логнормального распределения: $Median(\eta) = Mode(\eta) \cdot e^{\sigma^2}$.

Для обозначения моды, математического ожидания, медианы и параметра σ будем использовать соответственно переменные **mod**, **mean**, **med, sig**. Для чистоты эксперимента очистим их от возможных прежних значений: **Clear[mod, mean, med, sig]**.

Замечание. Имена очищаемых переменных могут содержать знак *****, при этом их следует заключать в кавычки. Например, **Clear[**"**m***"**]** очистит переменные **mod, mean** и **med, Clear[**"*****"**]** – сразу все переменные. Еще более радикальный способ начать расчеты с "чистого листа" – закрыть ядро (Kernel) пакета **MATHEMATICA**.

Выполним присвоения:

mod=500; mean=1000; med=mod E^sig^2.

Напомним, что в точке **med** графики плотностей, описывающих доход бедного (логнормальный закон) и богатого населения (половина от плотности распределения Парето) должны сомкнуться. Мы отказались от предположения, что медиана составляет 800 у.е. и пытаемся найти ее численное значение. Очевидно, что для этого достаточно найти значение переменной **sig**. Но сначала нужно исключить переменную α. Приравняем значения плотности (14.1) и уполовиненной плотности (14.4) при x, равном значению медианы. (x=a для (14.1) и x=c для (14.4)). Получим уравнение: $\frac{1}{\sigma\sqrt{2\pi}} = \frac{\alpha}{2}$, которое решим относительно α:

sol=Solve[1/(Sqrt[2 Pi] sig)==al/2,al].

(Синтаксис пакета требует в записи уравнений двойного равенства.)

Замечание. Знак умножения (*) в пакете **MATHEMATICA** можно заменить пробелом, что мы только что сделали. Вообще, что выгодно отличает язык **MATHEMATIC**и от универсальных языков программирования – это правильное восприятие естественных нотаций, например, **2x.**

Еще одно замечание. Мы могли избежать выписывания формул для плотностей следующим образом:

```
mylog=LogNormalDistribution[Log[med],sig]
myPar=ParetoDistribution[med,al]
sol=Solve[PDF[mylog,med]==0.5 PDF[myPar,med],al].
```

И еще одно замечание. MATHEMATICA позволяет использовать традиционное обозначение для греческих букв: **α** (набираем: Esc–a–Esc), **β** (Esc–b–Esc),…,**π** (Esc-p-Esc),…. Символ **π** обозначает известную константу (3.14….), остальные буквы могут использоваться для обозначения переменных.

Nota bene! Решение получается не численное (так как **al** неизвестно), а в виде *подстановки*, точнее – списка, хотя и состоящего из единственного элемента. (Дело в том, что **MATHEMATICA** не знала заранее, сколько корней будет получено.) Для упрощения изложения запишем этот первый и единственный элемент списка **sol** в переменную **s**:

s= sol[[1]]. Напомним, что **al** – величина *переменная*, поэтому при дальнейших обращениях к ней найденное только что *значение* будет утрачено. Чтобы зафиксировать его, введем новую переменную (**alpha**), в которую сделаем подстановку **s**. Синтаксис этой операции имеет вид:

alpha=al/.s.

Вычислим средний доход бедной части населения (как функцию от **sig**). Введем переменную:

mylog=LogNormalDistribution[Log[med],sig].

Согласно указанию к упражнению 14.3 получаем:

$$\text{meanpoor} = \int_0^{\text{med}} 2x\ PDF[\text{mylog}, x]\,\mathrm{d}x$$

Средний доход богатого населения отличается от общего среднего дохода на разность между последним и доходом бедного населения: **meanrich=mean+(mean-meanpoor**).

Таким образом, мы получили для среднего дохода богатого населения сложное выражение от **sig**. Вместе с тем, этот же средний доход может быть найден по формуле мат. ожидания распределения Парето (14.5), где параметр ***c*** совпадает со значением переменной **med**: **cPareto=med; meanPareto=alpha/(alpha-1)cPareto**.

Итак, мы нашли две разные формулы для выражения среднего дохода богатого населения. Рассмотрим их разность и найдем значения переменной

sig, при которых она обращается в 0. Пакет **MATHEMATICA** позволяет визуально определить количество нулей функции с помощью графика.

Введем функцию **f=meanPareto–meanrich**. Прежде чем строить ее график, найдем область ее определения. Построим систему ограничений на значения переменной **sig**, исходя из экономического смысла параметров. Подключим модуль для решения алгебраических неравенств:

```
<<Algebra`InequalitySolve`
```

и запишем функцию для решения неравенств:

```
InequalitySolve[mod<=med&&med<=mean&&sig>0&&alpha>1,sig
].
```

Здесь значок **&&** означает “логическое и”, то есть одновременное выполнение условий. Решением этого неравенства является интервал: $0 < \mathrm{sig} < \sqrt{\frac{2}{\pi}}$.

Построим на нем (исключая границы) график функции **f**

```
Plot[f,{sig,0.1,Sqrt[2/Pi]-0.1}].
```

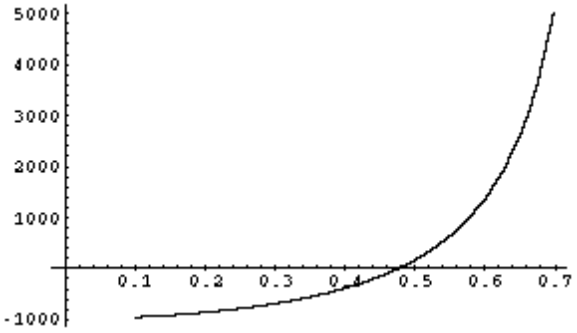


Рисунок 14.1. График функции **f**.

Очевидно, что уравнение **f=0** имеет единственный корень.

На графике видно, что единственный ноль функции **f** близок к 0.5 – возьмем эту точку в качестве начального приближения к решению: **solution=FindRoot[f==0,{sig,0.5}]**. В отличие от функции **Solve**, действующей по принципу “огласите весь список, пожалуйста”, то есть выдающей весь набор найденных решений, функция **FindRoot** находит только один корень, к которому сходится итерационный процесс, начатый в указанной точке. Получаем значение **sig**, равное 0.476 011. Вычислим теперь значения параметра **alpha** и медианы (точки сопряжения плотностей) с помощью найденного значения **sig**:

{alpha,med}={alpha,med}/.solution. Они соответственно равны 1.676 19 и 627.155.

Замечание. Запомните полученное значение **α=1.676 19.** Позднее мы еще вернемся к нему.

Построим графики обоих распределений с найденными параметрами:

```
mylog=mylog/.solution; g1=Plot[PDF[mylog,x],{x,0,med}].
```

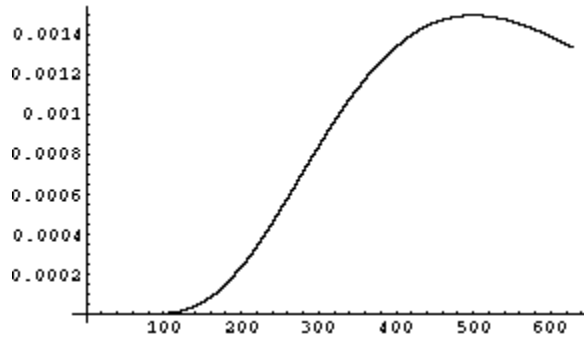


Рисунок 14.2. График плотности логнормального распределения. Описывает распределение дохода бедной части населения.

График будет храниться в переменной **g1**.

```
myPareto=ParetoDistribution[med, alpha],
g2=Plot[0.5 PDF[myPareto,x],{x,med,1500}]
```

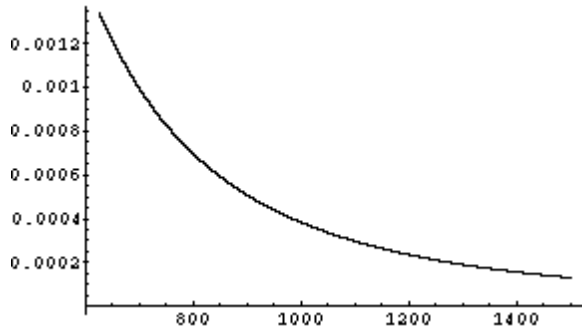


Рисунок 14.3. График плотности распределения. Парето. Описывает распределение дохода богатой части населения.

График будет храниться в переменной **g2**.

Теперь – самый ответственный момент. Изобразим оба графика на одной координатной плоскости: **Show[g1,g2]**.

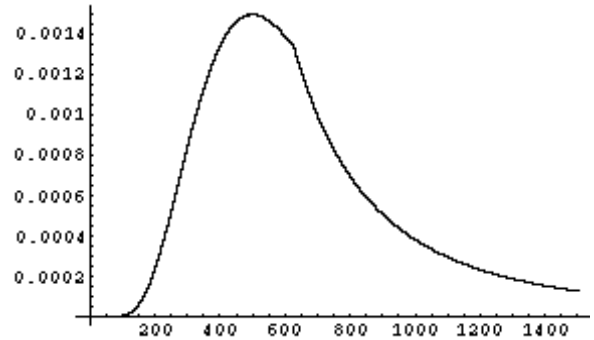


Рисунок 14.4. График плотности распределения дохода всего населения Вашего города.

Ура! Графики сомкнулись (рисунок 14.4), причем линия получилась достаточно гладкой, – это означает, что производные обеих функций мало различаются в точке стыковки.

Последнее и самое приятное замечание. Построение закона распределения дохода жителей Вашего города опирается на нахождение корня уравнения **f**=0. Без пакета **MATHEMATICA** доказательства существования и единственности решения было бы делом весьма непростым. Мы же поступили подобно древним египтянам, которые вместо математического доказательства рисовали на своих папирусах чертеж (рис. 14.1) и писали “смотри!”.

Итак, мы описали распределение дохода. Это позволит нам ответить на многие ***вопросы – причем не только математические, но и социально-экономические.***

Напомним, что в начале нашего небольшого исследования мы уже пытались оценить долю населения с фиксированным диапазоном дохода. При этом мы опирались на предположения о логнормальном распределении

дохода для всего населения и о том, что медиана распределения составляет 800 у.е. Поскольку полученные результаты нас не вполне удовлетворили (они противоречили нашим наблюдениям в отношении богатого населения), мы уточнили математическую модель – построили плотность распределения дохода как функцию, состоящую из двух "кусков": до медианы она совпадает с плотностью логнормального распределения, а после – с плотностью распределения Парето. Посмотрим, как согласуется с реальностью наша уточненная модель.

Для начала выясним, *сколько жителей городка беднее Вас*. Понятно, что это все, кто относится к категории бедных (50%) плюс некоторая часть богатого населения. Имеем:

$$P(\eta < 800) = \frac{1}{2} + \int_{med}^{800} \frac{1}{2} p_{\eta}^{(Парето)}(x)\,dx = \frac{1}{2} + \frac{1}{2} F_{\eta}^{(Парето)}(800) \quad (14.6)$$

Здесь $p_{\eta}^{(Парето)}(x)$ – плотность распределения Парето, $F_{\eta}^{(Парето)}(x)$ – функция этого распределения (с только что вычисленными значениями параметров).

Вычислим последнее слагаемое суммы (14.6) в пакете **MATHEMATICA**: **`CDF[myPareto,800]/2`**. Получаем 0.167 515. Таким образом, беднее Вас 50%+16.7%, то бишь две трети населения Вашего городка!

Теперь уточним *процент населения, имеющий доход выше среднего.* Для удобства изложения введем функцию $f(x)$, равную вероятности получения дохода, не меньшего x: $f(x) = P(\eta \geq x)$.

$$f(x) = 1 - P(\eta < x) = 1 - (\frac{1}{2} + \int_{med}^{x} \frac{1}{2} p_{\eta}^{(Парето)}(t)\,dt) = \frac{1}{2} - \frac{1}{2} F_{\eta}^{(Парето)}(x) \quad (14.7)$$

Для x=1 000 вычислим: **`0.5-0.5 CDF[myPareto, 1000]`**. Получаем: 0.228 735. Таким образом, доход выше среднего имеют 22,87% горожан (но никак не 37,24%, как предполагалось ранее).

Сколько же человек имеют очень маленький доход, не превышающий 400 у.е.? **`CDF[mylog,400]`** составляет 17,24%, что мало отличается от результата, полученного в начале нашего исследования (16%).

Какой процент населения имеет доход свыше 10 000? Согласно (14.7) получаем: **`0.5-0.5 CDF[myPareto,10000]`**, что составляет примерно 0,48%, то есть почти в 5 раз выше предыдущего значения. Вы были абсолютно правы в своих оценках высшего света!

Ну и наконец, выясним, *какой же доход является "проходным баллом" в высшее общество?* Найдем его (как значение переменной x) из условия: $p(\eta < x) = 0.99$. Получаем:

$$\frac{1}{2} + \frac{1}{2} F_{\eta}^{(Парето)}(x) = 0.99 \Rightarrow \frac{1}{2} F_{\eta}^{(Парето)}(x) = 0.49 \Rightarrow F_{\eta}^{(Парето)}(x) = 0.98 .$$

Корень этого уравнения является 98%-ым квантилем распределения Парето. **`Quantile[myPareto,0.98]`** есть 6 470.91, что чуть ли не вдвое выше предыдущего значения, в истинности которого Вы сильно сомневались.

Итак, 1% населения (составляющий высшее общество Вашего городка) имеет доход не ниже 6 470 у.е.; подняв планку до 10 000 у.е., мы "отсеем"

0,52% всего населения, то есть 52% высшего света (составляющего, как уже было сказано, 1% общества). Проверим высший свет на прочность. Посмотрим, что от него останется, если поднять планку еще во столько же раз (то есть примерно в 1.55).

Выведем формулу "отсева" при изменении минимального дохода с величины ***x*** на ***y***. Поскольку "отсев" составляет $f(x) - f(y)$, относительная величина "отсева" равна $\frac{f(x)-f(y)}{f(x)}$. Используя формулу (14.7), проделаем вычисления в пакете **MATHEMATICA**:

```
z=0.5-0.5 CDF[myPareto,10000];
w=0.5-0.5 CDF[myPareto,15500]; (z-w)/z.
```

Получаем 0.520 32, то есть те же самые 52%.

Упражнение 14.4. Убедитесь, с помощью пакета **MATHEMATICA**, что **относительная** величина "отсева" при увеличении суммы дохода на 1% постоянна и равна параметру ***α*** распределения Парето.

Указание. Положите переменную ***x*** равной некоторому значению дохода, например, 1 000. Тогда ***y*** составит 1,01 ***x***, то есть 1 010. Проведите эксперименты для разных значений ***x***.

Пусть ***N(x)*** – количество людей с доходом, не меньшим ***x***. Ясно, что $\frac{f(x)-f(y)}{f(x)} = \frac{N(x)-N(y)}{N(x)}$. Таким образом, параметр ***α*** характеризует *эластичность* ***N(x)*** по доходу ***x***.

Нетрудно показать, что

$$N(x) = n\left(\frac{c}{x}\right)^{\alpha}, \qquad (14.8)$$

где *n* – общее число жителей.

Упражнение 14.5. С помощью формулы для функции распределения Парето (см. упражнение 14.2) докажите равенство (14.8).

Итак, абсолютная величина "отсева" пропорциональна ***N(x)***, а значит, обратно пропорциональна x^{α}. Вспомним, что ***x*** – это доход, а **α=1.676 19**. Значит, при поднятии планки на 1% "отсеяно" будет тем меньше людей, чем больше ***x***. Такая "непотопляемость" богатых подтверждает известную народную мудрость "деньги к деньгам", причем при больших значениях ***α*** эта тенденция проявляется ярче.

По мнению последователей Парето, значение ***α***, близкое к 1,5, характеризует состояние социального равновесия; увеличение ***α*** свидетельствует о социальной нестабильности (вследствие большой поляризации общества). Таким образом, как поется в популярной некогда песне, "городок наш ничего", хотя некоторое превышение ***α*** над идеальным значением все же заставляет задуматься… Заметьте, все началось с праздного, казалось бы, любопытства относительно чужих доходов, а в итоге мы получили инструмент для экономического и социологического анализа.

Приложение. Список использованных возможностей пакета MATHEMATICA.

Ключ	Действие
Shift+Enter	Завершение ввода строки; отправка на выполнение.
F1	Вызов HELP.
Cell	Обращение к меню для работы с ячейками.
Palettes	Обращение к меню для работы с палитрами.
Ctrl+^	Возведение (выделенного выражения) в степень.
Ctrl + пробел	Возвращение курсора в основную строку.
Ctrl +% (Ctrl+5)	Перемещение курсора между нижним и верхним пределами интегрирования.
Ctrl +K +подстрока	Вызов списка ключевых слов, содержащих данную подстроку.
??подстрока	Вызов списка ключевых слов, содержащих данную подстроку. В подстроке для замещения отсутствующих символов допустимо использование символа *.
(Esc–a–Esc)	Символ α.
(Esc–b–Esc)	Символ β.
. . . .	. . .
(Esc–p–Esc)	Символ π.
. . .	. . .
Команда	**Действие**
Out[номер]	Обращение к выходной ячейке с указанным номером.
%	Обращение к предыдущей выходной ячейке
N[выражение, целое число]	Преобразование выражения в 10-ичную дробь. Необязательный параметр – целое число – общее количество знаков результата.
Clear	Очистка переменных.
/.	Операция подстановки.
Limit	Нахождение предела.
Plot	Построение графика.
Show	Отображает графики, являющиеся аргументами, на одной координатной плоскости.
D	Дифференцирование.
Integrate	Интегрирование.
FindRoot	Нахождение корня уравнения.

Solve	Нахождение всех корней уравнения.
<<Statistics `Master`	Вызов модуля статистических расчетов.
PDF	Вычисление плотности распределения.
CDF	Вычисление функции распределения.
Mean	Вычисление математического ожидания.
Quantile	Вычисление квантиля распределения.
<<Algebra `InequalitySolve`	Вызов модуля решения алгебраических неравенств.

6 семестр

Лабораторная работа № 1.

Тема: Использование Mathcad в качестве суперкалькулятора.

Пример 1. Для набора выражения используем обычную математическую нотацию:

$1/\sqrt{2} = 0.707$. Знак квадратного корня мы найдем, раскрыв арифметическую палитру. В конце выражения поставим знак равенства “=”

$\frac{1}{\sqrt{2}} = 0.707$

Пример 2. Можно присвоить значения переменным: $a:=1$ $b:=2.35$ $p:=\pi$

Ввод заканчивается клавишей Enter или щелчком мыши вне определения.

Здесь мы обозначили переменные буквами: *a*, *b, p;* но можно использовать произвольный набор символов для обозначения переменных. Имена переменных чувствительны к регистру. Вначале вводится имя переменной, затем символ “:” (или знак =), затем число или выражение (в частности, мы использовали предопределенную константу π из арифметической пары, Ctrl+p). Синий уголок показывает текущий операнд выражения, он может быть расширен клавишей **“Пробел”.** *Обратите внимание, что в качестве разделителя целой и дробной части числа используется точка*. Теперь этими переменными можно пользоваться при арифметических вычислениях.

:= это оператор присваивания, = это команда «Вывести значение».

$\frac{a+b}{2} = 1.675$ $c := \frac{a-b}{2}$ $d := \sin\left(\frac{p}{2}\right)$ c=-0,675 d=1

Сейчас видна разница в использовании оператора присваивания := и знака =.

Пример 3. Вычислите для x=1 следующие функции:

$y := \frac{1}{\sqrt[3]{x^2} + \sqrt[6]{x^5}}$ $y := \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ $y := \frac{\mathrm{atan}(x)^2}{2}$. Функция arctg(x) обозначена здесь как atan(x). Набирается «вручную».

Вычислите значения при x=3, а также при x=5.

Для корректного ввода формул необходимо пользоваться арифметической палитрой и кнопкой **«Вставить функцию»** и копировать формулы, используя кнопки панели инструментов.

Правила видимости: значение переменной доступно правее и ниже ее определения.

Глобальные переменные доступны везде на рабочем листе и вводятся знаком ~, например, введем N ~100, получим N≡100

Если вы хотите изменить количество знаков результата вычислений после десятичной точки, это можно сделать в меню **Format\Number…\Displayed Precision(3)** или просто дважды щелкнуть мышкой по выражению, после чего заменить 3 на 6.

Установим, например, для значения выражения 6 значащих цифр: $\frac{a}{b} = 0.425532$

Для ввода текстового комментария необходимо ввести знак двойной кавычки “, затем вводить текст. При достижении конца строки происходит автоматический перенос на следующую. Текстовая область, как и любая другая, может быть перемещена на рабочем листе или скопирована в буфер. Маркеры текстовой области позволяют менять ее размеры.

Упражнения для самостоятельной работы. Вычислите при x=2:

$$y1 := \frac{2.087 \cdot x^3 + 3.24 \cdot \sqrt[3]{x}}{1 + \sqrt{x}} \qquad y2 := \frac{1}{\pi} \cdot \cos\left(\frac{x}{\pi}\right)^2 - \frac{1}{2 \cdot \pi} \cdot \sin\left(\frac{x}{2 \cdot \pi}\right) \qquad y3 := \frac{\sqrt{1 - \sin(a \cdot x)^2}}{b - p \cdot \tan(x)}$$

Числовые массивы. Матрицы.

Матрица – прямоугольная таблица чисел (массив). Поэтому будем понимать эти термины как синонимы. В Mathcad реализованы одно- и двумерные матрицы, причем одномерные матрицы – это просто массивы у которых один столбец. Создаются матрицы при помощи кнопки палитры инструментов или команды **Insert\Matrix…**, где указывается количество строк, столбцов **Rows и Columns**. Для примера создадим матрицу размером 3*3 и 3*1:

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 8 & 9 & 4 \\ 7 & 6 & 5 \end{pmatrix} \qquad A := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Матрицу с одним столбцом называют вектор-столбец.

Принято обозначать матрицы большими латинскими буквами.

С матрицами можно проделывать множество операций, имеется даже специальная матричная алгебра, но мы ограничимся лишь обычными операциями с массивом чисел:

1. *Умножение матрицы на число*. G:=2·A $\quad G = \begin{pmatrix} 2 & 4 & 6 \\ 16 & 18 & 8 \\ 14 & 12 & 10 \end{pmatrix}$

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot 0.5 = \begin{pmatrix} 0.5 \\ 1 \\ 1.5 \end{pmatrix}$$

2. *Сложение матриц*. $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 0.5 \\ 1 \\ 1.5 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 3 \\ 4.5 \end{pmatrix}$

C:=A+2·A $\quad C = \begin{pmatrix} 3 & 6 & 9 \\ 24 & 27 & 12 \\ 21 & 18 & 15 \end{pmatrix}$

Примечание. Совершенно очевидно, что в операциях сложения размеры матриц должны совпадать.

Доступ к элементам матриц.

Имея дело с массивами чисел неплохо было бы иметь возможность извлечения отдельного числа из матрицы. Для этого реализован механизм индексирования. Так в одномерной матрице (вектор-столбец) все значения пронумерованы от 0 до n-1, где n количество значений. Обращение к элементу массива производится по индексу. Например, в матрице B три значения с индексами 0, 1, 2 и обращение к ним производится , как к переменной с индексом:

$B_0=1$ $B_1=2$ $B_2=3$ Индекс вводится символом квадратной скобки [- B[0, B[1, B[2 или из арифметической палитры.

Примечание. *Переменная с индексом может присутствовать в арифметическом выражении наряду с другими переменными.*

Обращение к двумерному массиву производится аналогично, только приходится указывать два индекса через запятую: первый индекс – это номер строки, второй – номер столбца. Как и ранее нумерация начинается с 0. Например:

$A_{0,0}=1$ $A_{0,2}=3$ $A_{2,0}=7$ $A_{2,2}=5$

Упражнения для самостоятельной работы. Введите матрицы размером 2*2, 3*3, 4*4, 1*3 с произвольным набором чисел.

Что касается последней матрицы размером 1*3, то хотя и одномерная матрица – вектор-строка, доступ к ее элементам возможен только как к двумерной, где первый индекс равен 0, например:

Z:= (1 2 3) $Z_{0,0}=1$ $Z_{0,1}=2$ $Z_{0,2}=3$

Примечание. Элементами матрицы могут быть и арифметические выражения.

Лабораторная работа № 2.

Тема: Функции. Интервальная переменная.

Определить функцию в Mathcad достаточно просто, для этого необходимо ввести имя функции, в скобках её параметры и оператор присваивания. После чего вводится арифметическое выражение. Затем функция может использоваться наравне с встроенными функциями. Введем для примера параболическую функцию:

***a*:=1 *b*:=1 *c*:=-1** Нам пришлось предварительно описать три константы *a*, *b*, *c*, $f(x) := a \cdot x^2 + b \cdot x + c$ иначе функция не может быть вычислена.

Теперь, для того чтобы получить значение функции, достаточно записать:

***f(0)*=-1** ***f*(1.5)=2.75** и так далее.

Однако, если бы этим ограничивались возможности Mathcad, то этот пакет так бы и остался большим калькулятором. К счастью был предусмотрен механизм многократного повторения вычислений, нечто подобное циклам в языках программирования. Введено понятие интервальной переменной в формате:

var*:=начальное значение[, начальное значение+шаг]..конечное значение*

в скобках указан необязательный параметр шаг, по умолчанию равный 1.

Двоеточие ".." вводится клавишей точка с запятой ":" или кнопкой арифметической палитры

Введем для примера интервал изменения аргумента x на отрезке **[-2;2]** с шагом **h=0.1**.

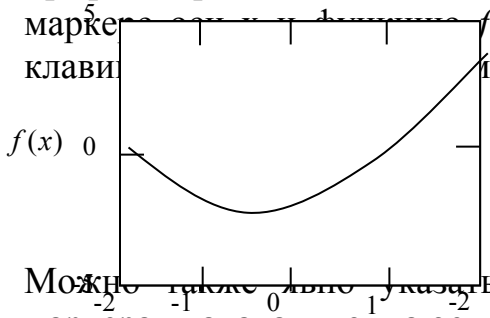
$x := -2, -1.9 .. 2$ Фактически мы получаем набор из 41 значения аргумента. Чтобы убедиться в этом, достаточно ввести x=

Для того чтобы вывести таблицу значений функции, введите ***f*(x)** и знак "=", вы получите значения функции.

Таким образом можно увидеть только первые 50 значений.

Ну, а сейчас можно построить график. Воспользуемся графической палитрой, раскрыв которую выберем **x-y график**.

График строится довольно просто, нужно только указать x переменную в маркере оси x и функцию *f(x)* в маркере оси y. Заканчивается построение клавишей мыши вне графика.



Можно также явно указать начальное и конечное значение по осям в маркерах начала и конца оси, иначе они определятся автоматически.

Выделив график двойным щелчком мыши, можно произвести его настройку, в частности, определить тип, цвет и толщину линии, а также выбрать оси.

Упражнение для самостоятельной работы. Постройте графики функций:

$u(x) := x \cdot \sin(x)$ $v(x) := \sqrt{1 + \sin(x)^2}$ $w(x) := e^{-x^2}$ $z(x) := \log(1 + \sqrt{x - 1})$

Диапазон для изменения аргумента по умолчанию будет **[-10;10].** Шаг принять равным **0,2.**

Примечание: на одном х-у графике можно построить до 16 кривых. *Функции вводятся через **запятую***.

Матрицы и переменные с индексом

К сожалению, использование интервальной переменной для построения функции вызывает определенные трудности. Это связано с тем, что при каждом обращении к интервальной переменной система производит многократные вычисления и, зачастую, совершенно напрасно пересчитывает одни и те же значения. Чтобы избежать лишней работы проще всего вычислить таблицу значений функции один раз, а затем обращаться к табличным значениям.

Для реализации данной задачи используется интервальная переменная и переменная с индексом. Например, если вернуться к предыдущей задаче, можно решить её следующим образом:

введем индекс, как интервальную переменную **i:=0..40**

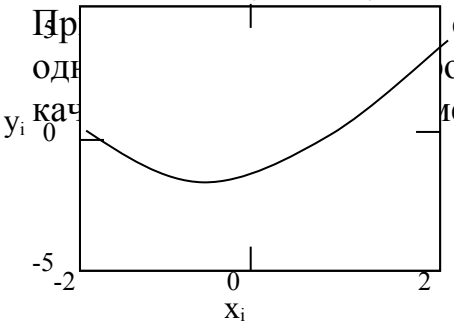
введем переменную с индексом, которая будет меняться

от -2 до 2 с шагом 0.1, для описания переменной **x_i:=-2+0.1·i**

Если вычислить **y_i:=*f*(x_i)** мы получим вектор-столбец значений функции.

Введите х= и у=, получите эти матрицы.

Пр с индексом мы создали две одномерные матрицы с одн окой. Теперь можно вновь построить график, где в кач мента мы и укажем эти переменные с индексом.



Аналогично можно использовать двумерную матрицу для построения функции двух переменных, например, определим функцию:

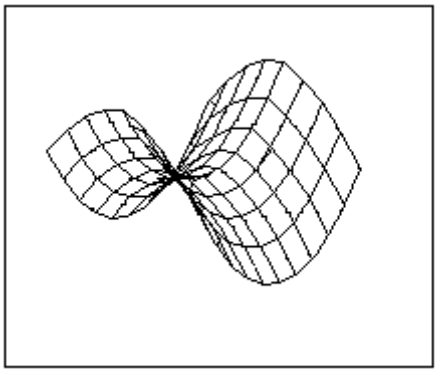
f(x,y):=x^2-y^2 и две интервальных переменных ***i*:=0..10** ***j*:=0..10**

Теперь определяем две переменные с индексом ***x_i:=-5+i***

y_j:=-5+j

Определим двумерную матрицу: ***$M_{i,j}$:=f(x_i,y_j)*** и построим поверхность

В качестве единственного аргумента графика указываем имя матрицы *M*



M

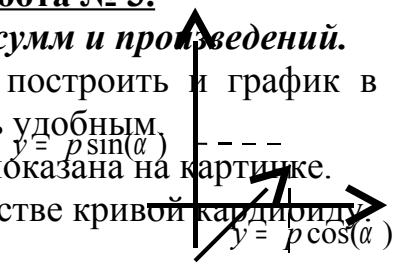
M=

	0	1	2	3	4	5	6	7	8	9
0	0	9	16	21	24	25	24	21	16	9
1	-9	0	7	12	15	16	15	12	7	0
2	-16	-7	0	5	8	9	8	5	0	-7
3	-21	-12	-5	0	3	4	3	0	-5	-12
4	-24	-15	-8	-3	0	1	0	-3	-8	-15
5	-25	-16	-9	-4	-1	0	-1	-4	-9	-16
6	-24	-15	-8	-3	0	1	0	-3	-8	-15
7	-21	-12	-5	0	3	4	3	0	-5	-12
8	-16	-7	0	5	8	9	8	5	0	-7
9	-9	0	7	12	15	16	15	12	7	0
10	0	9	16	21	24	25	24	21	16	9

Лабораторная работа № 3.

Тема: Графики. Вычисление сумм и произведений.

В Mathcad, кроме x-y – графика можно построить и график в полярных координатах, что иногда оказывается очень удобным.



Связь полярных и декартовы x координат показана на картинке.

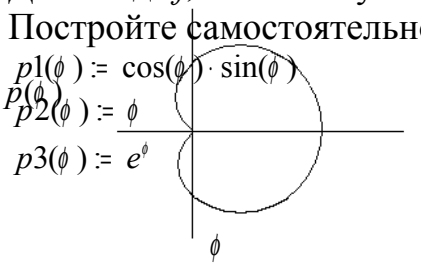
Построим полярный график, выбрав в качестве кривой кардиоиду:

$p(\phi) := 1 + \cos(\phi)$ $\phi := 0,0.1..2\cdot\pi$

Для ввода *f, r* используем палитру греческих символов.

Постройте самостоятельно графики функций:

1. $p1(\phi) := \cos(\phi)\cdot\sin(\phi)$
2. $p2(\phi) := \phi$
3. $p3(\phi) := e^{\phi}$



Для построения столбиковой диаграммы необходимо задать матрицу значений

$$D := \begin{pmatrix} 5 & 8 & 1 \\ 5 & 4 & 2 \\ 5 & 2 & 3 \end{pmatrix}$$

Как видно из графика, каждая колонка матрицы создает ряд значений.

Постройте столбиковые диаграммы для произвольного набора чисел, например:

Примечание. Просмотрите примеры в электронных книгах центра ресурсов.

Вычисление сумм и произведений

Воспользуемся палитрой вычислений  .

Например: $\sum_{n=1}^{100} \frac{1}{n^2} = 1.635$ $\sum_{n=1}^{20} \frac{(-1)^n}{(2\cdot n)!} = 0.54$ $\sum_{k=1}^{1000} \frac{1}{(2\cdot k-1)\cdot(2\cdot k+1)} = 0.5$

По определению, n!=1*2*…*n , т.е. произведение всех чисел до n включительно.

Здесь мы использовали значок суммы с указанием границ суммирования.

Значок суммирования только с указанием индекса используется в тех случаях, когда пределы изменения индекса заданы переменной интервального типа.

Например: $X := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ i:=0..2 $\sum_i X_i = 6$

n:=1..100 u(n):=$n^2+2\cdot n+1$ $\sum_n \frac{1}{u(n)} = 0.635$

Аналогично вычисляются произведения. По определению: $\prod_{i=1}^{n} (a_i = a_1 \cdot a_2 \cdot ... a_n)$

Например:

$\prod_{k=2}^{10000}\left(1-\frac{1}{k^2}\right) = 0.5$ $\prod_{n=1}^{10000}\left[1+\frac{(-1)^{n+1}}{2\cdot n-1}\right] = 1.414$ x:=0.5 $\prod_{k=0}^{1000}\left(1+x^{2^k}\right) = 2$

Вычислим для матриц M и D, которые мы использовали для построения объемных графиков, сумму и произведение.

i:=0..2 $\sum_i M_{i,0} = 25$ $\prod_i M_{i,1} = 0$ $\sum_i D_{i,i} = 12$ $\prod_i D_{i,i} = 60$

Символьные вычисления

Суммы и произведения можно вычислять в символьном виде, для этого, вместо оператора “=”, необходимо воспользоваться кнопкой математической палитры, например:

$\sum_{n=1}^{\infty} \frac{1}{n^2} \rightarrow \frac{1}{6}\cdot\pi^2$ $\sum_{n=1}^{\infty} \frac{1}{e^n} \rightarrow \frac{1}{(-1+\exp(1))}$

$\sum_{n=0}^{\infty} \frac{(-1)^n\cdot z^{2\cdot n}}{(2\cdot n)!} \rightarrow \cos(z)$ $\sum_{n=0}^{\infty} \frac{(-1)^n\cdot z^{2\cdot n+1}}{(2\cdot n+1)!} \rightarrow \sin(z)$

$\sum_{n=0}^{7} \frac{(-1)^n\cdot z^{2\cdot n}}{(2\cdot n)!} \rightarrow 1-\frac{1}{2}\cdot z^2+\frac{1}{24}\cdot z^4-\frac{1}{720}\cdot z^6+\frac{1}{40320}\cdot z^8-\frac{1}{3628800}\cdot z^{10}-\frac{1}{479001600}\cdot z^{12}-\frac{1}{871782912}$

Получаем просто ряд из нескольких слагаемых. Это значит, система не смогла упростить выражение.

Лабораторная работа № 4.

Тема: Операции с матрицами.

При решении задач линейной алгебры часто приходится проводить операции с матрицами. При работе с матрицами необходимо использовать панель "Векторы и матрицы" (Vector and Matrix Toolbar). Кроме того, часто используются следующие встроенные функции:

matrix(m,n,f) – создает и заполняет матрицу размерности m*n , элемент которой, расположенный в i-той строке, j-том столбце, равен значению f(i,j) функции f(x,y);

diag(v) - создает диагональную матрицу , элементы главной диагонали которой хранятся в векторе v;

identity(n) - создает единичную матрицу порядка n;

augment(A,B) - к матрице A приписывает справа матрицу B (матрицы A и B должны иметь одинаковое число строк);

stack(A,B) - к матрице A приписывает снизу матрицу B (матрицы A и B должны иметь одинаковое число столбцов);

submatrix(A,ir,jr,ic,jc) - выделяет из матрицы A подматрицу, расположенную в строках с ir по jr и в столбцах с ic по jc;

При работе с матрицами обычно принято нумеровать строки и столбцы, начиная с 1 (а с MathCad-е нумерация по умолчанию начинается с нуля, поэтому рекомендуется для удобства восприятия матричных вычислений выполнить команду: **ORIGIN:=1**

$f(x,y) := x + y$ $\quad A := \text{matrix}(3,4,f)$ $\quad A = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{pmatrix}$ $\quad i := 1..4 \quad v_i := i$

$B := \text{diag}(v)$ $\quad E := \text{identity}(3)$

$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$ $\quad E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $\quad D := \text{augment}(A,E)$

$D = \begin{pmatrix} 0 & 1 & 2 & 3 & 1 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 & 0 \\ 2 & 3 & 4 & 5 & 0 & 0 & 1 \end{pmatrix}$

$F = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$ $\quad F := \text{stack}(A,B)$ $\quad G := \text{submatrix}(F,4,5,1,2)$

$G = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$

Для вычисления различных числовых характеристик матриц используются следующие функции:
last(v) - вычисление номера последней компоненты вектора v ;
length(v) - вычисление количества компонент вектора v;
rows(A) - вычисление числа строк в матрице A;
cols(A) - вычисление числа столбцов в матрице A;
max(A) - вычисление максимального элемента в матрице A;
min(A) - вычисление минимального элемента в матрице A;
tr(A) - вычисление следа квадратной матрицы A (т.е. суммы ее диагональных элементов);
rank(A) - вычисление ранга матрицы A;

norm1(A) - первая норма матрицы $||A||_1 = \max_j \sum_{i=1}^{n} |a_{i,j}|$ (т.е. максимальная сумма модулей чисел по столбцам матрицы);
norm2(A) - вторая норма матрицы $||A||_2 = \sqrt{\lambda_{\max}(A \cdot A^T)}$, где $\lambda_{\max}(A \cdot A^T)$ - максимальное собственное значение матрицы $A \cdot A^T$.
norme(A) - евклидова норма матрицы $||A||_e = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{i,j}|^2}$

normi(A) - $\max_i \sum_{j=1}^{n} |a_{i,j}|$, (т.е. максимальная сумма модулей чисел по строкам матрицы);
Выполните следующие действия:

last(V) = 4 length(V) = 4

rows(D) = 3 cols(D) = 7 max((D) = 5 tr(B) = 10
rank(A) = 2

norm1(B) = 4 norm2(B) = 4 normi(B) = 4 norme(B) = 5.477

При численном решении задач линейной алгебры используются следующие **функции:**

rref(A) – приведение матрицы к ступенчатому виду с единичным базисным минором (выполняются элементарные операции над строками матрицы);
eigenvals(A) – вычисление собственных значений квадратной матрицы A:
eigenvecs(A) – вычисление собственных векторов квадратной матрицы A: значением функции является матрица, столбцы которой есть собственные векторы матрицы A (порядок их следования отвечает порядку следования собственных значений, вычисленных функцией **eigenvals(A)**;

eigenvecs(A,I) – вычисление того собственного вектора матрицы A, который соответствует I – тому собственному значению матрицы;
lsolve(A,b) – решение системы линейных алгебраических уравнений Ax = b

$$C := \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 1 \end{pmatrix} \qquad \text{eigenvals}(C) = \begin{pmatrix} 0.271 \\ -2.71 \end{pmatrix} \qquad L = \begin{pmatrix} 0.271 \\ -2.71 \end{pmatrix}$$

$$\text{eigenvecs}(C) = \begin{pmatrix} 0.321 & -0.611 & -0.432 \\ 0.711 & 0.222 & 0.801 \end{pmatrix}$$

$$\text{eigenvec}(C, L_2) = \begin{pmatrix} 0.611 \\ -0.222 \end{pmatrix} \qquad b := \begin{pmatrix} 8 \\ 14 \end{pmatrix} \qquad X = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Для транспонирования матрицы (т.е. перестановки строк со столбцами, когда первая строка становится первым столбцом, вторая - вторым столбцом и т.д.) можно воспользоваться панелью "Матрица" ("Matrix") , выбрав значок $\mathbf{M^T}$. Чтобы вычислить определитель матрицы, можно воспользоваться на этой панели значком **|x|**. Для вычисления обратной матрицы необходимо применить значок $\mathbf{x^{-1}}$ (при этом следует помнить, что обратная матрица вычисляется только для квадратной матрицы с ненулевым определителем).

$|C| = -4$

$$C^{-1} = \begin{pmatrix} -1.5 & 2 & 0.5 \\ 0.75 & -0.5 & -0.25 \end{pmatrix} \qquad C^T = \begin{pmatrix} 2 & 1 & 4 \\ 3 & 2 & 1 \end{pmatrix}$$

Решение системы линейных уравнений методом Крамера
При решении задач линейной алгебры для вычисления неизвестных в системе линейных алгебраических уравнений можно воспользоваться методом Крамере, основанном на вычислении определителя. Если определитель $\Delta = \det A$ матрицы системы уравнений **Ax=b** отличен от нуля, тосистема имеет единственное решение $x_1, x_2, x_3, \ldots x_n$, определяемое по формулам Крамера $x_i = \Delta_i/\Delta$, где $\tilde{\Delta}_i$определитель матрицы n -ного порядка, полученной из матрицы системы заменой i - того столбца столбцом правых частей **b**.
Пример . Решение СЛАУ методом Крамера.

$$A := \begin{pmatrix} -1 & 2 & -3 & 4 \\ 0 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 3 \\ 10 \end{pmatrix} \qquad \Delta := |A|$$

$$\Delta 1 := \left| \begin{pmatrix} 30 & 2 & 3 & 4 \\ 10 & 2 & -1 & 1 \\ 3 & 1 & 1 & 1 \\ 10 & 1 & 1 & 1 \end{pmatrix} \right| \qquad \Delta 2 := \left| \begin{pmatrix} 1 & 30 & 3 & 4 \\ -1 & 10 & -1 & 1 \\ 0 & 3 & -1 & 1 \\ 1 & 10 & 1 & 1 \end{pmatrix} \right| \qquad \Delta 3 := \left| \begin{pmatrix} 1 & 2 & 30 & 4 \\ -1 & 2 & 10 & 1 \\ 0 & 1 & 3 & 1 \\ 1 & 1 & 10 & 1 \end{pmatrix} \right|$$

$$\Delta 4 := \left| \begin{pmatrix} 1 & 2 & 3 & 30 \\ -1 & 2 & -3 & 10 \\ 0 & 1 & -1 & 3 \\ 1 & 1 & 1 & 10 \end{pmatrix} \right|$$

Определитель отличен от нуля, следовательно, система имеет единственное решение

$\Delta = -4 \qquad \Delta 2 = -8 \qquad \Delta 3 = -12 \qquad \Delta 4 = -16$

$$x1 := \frac{\Delta 1}{\Delta} \qquad x2 := \frac{\Delta 2}{\Delta} \qquad x3 := \frac{\Delta 3}{\Delta} \qquad x4 := \frac{\Delta 4}{\Delta}$$

$x1 = 1 \qquad x2 = 2 \qquad x3 = 3 \qquad x4 = 4$

Эту задачу можно решить другим способом, используя понятие обратной матрицы:

$$x := A^{-1} \cdot b \qquad x = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$$

Для сравнения найдем решение системы Ax=b с использованием функции lsolve(A,b) :

$$x = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$$

Решение СЛАУ методом Гаусса

Метод Гаусса (или метод Гауссовых исключений) состоит в том, что систему уравнений сначала приводят к ступенчатому виду (прямой ход метода Гаусса), а затем находят неизвестные X (обратный ход). В MathCad прямой и обратный ходы метода Гаусса выполняет функция **rref(Ar)**. При этом матрица **Ar** представляет собой расширенную матрицу - к матрице коэффициентов при неизвестных **A** добавлен справа столбец свободных членов **b:**

$$Ag = \begin{pmatrix} 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 4 \end{pmatrix} \qquad Ar = \begin{pmatrix} -1 & 2 & -3 & 4 & 10 \\ 0 & 1 & -1 & 1 & 3 \\ 1 & 1 & 1 & 1 & 10 \end{pmatrix}$$

Как видно из решения, функция **rref(Ar)** привела систему уравнений к ступенчатому виду с единичной матрицей в первых столбцах. Последний столбец полученной матрицы и есть искомое решение. Для того, чтобы его найти, воспользуемся функцией **submatrix**:

$$x = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$$

Выполните самостоятельно.

Исследуйте и, если решение существует, найдите по формулам Крамера, Гаусса и функции **lsolve** решение системы Ax=b

$$A := \begin{pmatrix} -0.090 & -0.033 & 0.0067 & -0.098 \\ 0.150 & 0.033 & 0.050 & 0 \\ 2.857 & 0.100 & -0.300 & 0.025 \end{pmatrix} \qquad b := \begin{pmatrix} -0.098 \\ 0.183 \\ -0.041 \end{pmatrix}$$

Лабораторная работа № 5.
Тема: Дополнительные функции MathCad.

Числовые функции с условиями сравнения:

ceil(x) - наименьшее целое, большее или равное **x**

floor(x) - наибольшее целое, меньшее или равное **x**

mod(x,y) - остаток от деления **x/y** со знаком **x**

Функция условных выражений **if**

if (Условие, Выражение 1, Выражение 2) - если в этой функции условие выполняется, то будет вычисляться выражение1, в противном случае - выражение 2.

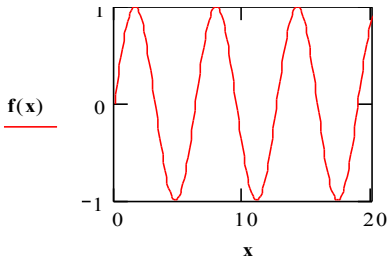
Пример

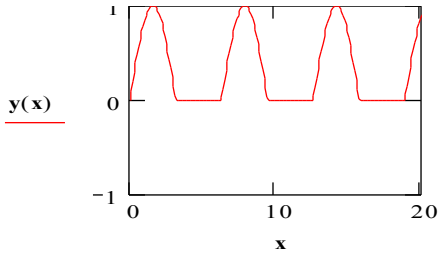
Синусоидальный сигнал

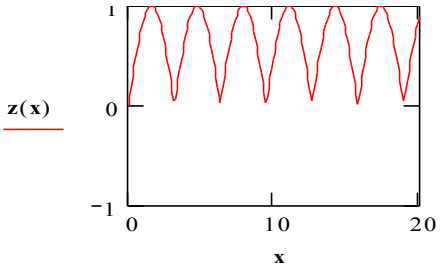
Сигнал однополупериодного выпрямления

Сигнал двухполупериодного выпрямления

Графики функций







Дополнительные матричные функции

lu(M) - выполняет треугольное разложение матрицы M: L и U -соответственно нижняя и верхняя треугольная матрицы.

qr(M) - выполняет ортогональное разложение матрицы **M: q** и **r** - соответственно ортогональная и верхняя треугольная матрицы

$$\mathbf{a} := \begin{pmatrix} 6 & 5 & 3 \\ 9 & 7 & 0 \end{pmatrix} \qquad \mathbf{lu}(\mathbf{a}) = \begin{pmatrix} 1 & 0 & 0 & 0.333 & 1 & 0 & 0 & 1.333 & 0 \\ 0 & 0 & 1 & 1.5 & -0.375 & 1 & 0 & 0 & -4.5 \end{pmatrix}$$

$$\mathbf{l} = \begin{pmatrix} 0.333 & 1 & 0 \\ 1.5 & -0.375 & 1 \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} 0 & 1.333 & 0 \\ 0 & 0 & -4.5 \end{pmatrix}$$

Проверка

$$\mathbf{l} \cdot \mathbf{u} = \begin{pmatrix} 2 & 3 & 1 \\ 9 & 7 & 0 \end{pmatrix}$$

$$\mathbf{qr}(\mathbf{a}) = \begin{pmatrix} 0.545 & -0.064 & -0.836 & 0 & -1.414 & -1.157 \\ 0.818 & 0.257 & 0.514 & 0 & 0 & -2.314 \end{pmatrix}$$

$$\mathbf{q} = \begin{pmatrix} 0.545 & -0.064 & -0.836 \\ 0.818 & 0.257 & 0.514 \end{pmatrix} \qquad \mathbf{r} = \begin{pmatrix} 0 & -1.414 & -1.157 \\ 0 & 0 & -2.314 \end{pmatrix}$$

Проверка

$$\mathbf{q} \cdot \mathbf{r} = \begin{pmatrix} 6 & 5 & 3 \\ 9 & 7 & 0 \end{pmatrix}$$

Функции сортировки для векторов и матриц

sort(v) - сортировка элементов векторов в порядке возрастания их значений

reverse(v) - перестановка элементов (после sort) в обратном порядке

csort(M,n) - перестановка строк матрицы M таким образом, чтобы отсортированным оказался n-ный столбец;

rsort(M,n) - перестановка столбцов матрицы M таким образом, чтобы отсортированной оказалась n - ная строка.

$$\mathbf{V} := \begin{pmatrix} 2 \\ 4 \\ 6 \\ 1 \end{pmatrix} \qquad \mathbf{VS} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 6 \end{pmatrix} \qquad \mathbf{VR} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Исходная матрица Сортировка по первому столбцу и по первой строке

$$\mathbf{M} := \begin{pmatrix} 3 & 15 & 27 \\ 17 & 2 & 13 \end{pmatrix} \quad \mathbf{csort}(\mathbf{M}, 1) = \begin{pmatrix} 1 & 12 & 4 \\ 3 & 15 & 27 \end{pmatrix} \quad \mathbf{rsort}(\mathbf{M}, 1) = \begin{pmatrix} 3 & 15 & 27 \\ 17 & 2 & 13 \end{pmatrix}$$

Решение систем линейных алгебраических уравнений методом простых итераций

Исходная системы уравнений Cx=d преобразуется к виду:

X=b + Ax и ее решение вычисляется как предел последовательности

$X^{(k)} = b + Ax^{(k-1)}$, k=1,2,….

Преобразовать систему Cx=d к виду x=b+Ax можно, выделив диагональные элементы:

$$x_i = \frac{1}{c_{i,i}}(d_i - \sum_{i \neq j} c_{i,j} \cdot x_j)$$

В качестве условия окончания итерационного процесса можно взять условие $\frac{\left\|x^{(k)} - x^{(k-1)}\right\|}{\left\|x^{(k)}\right\|} \leq \varepsilon$, где e – заданная погрешность приближенного решения

Пример Решить методом простых итераций систему уравнений:

$100x_1 + 6x_2 - 2x_3 = 200$
$6x_1 + 200x_2 - 10x_3 = 600$
$x_1 + 2x_2 - 100x_3 = 500$

Преобразованная система имеет вид:
$x_1 = 2 - 0.06x_2 + 0.02x_3$
$x_2 = 3 - 0.03x_1 + 0.05x_3$
$x_3 = 5 - 0.01x_1 - 0.02x_2$

$$b := \begin{pmatrix} 3 \\ 5 \end{pmatrix} \qquad A := \begin{pmatrix} -0.03 & 0 & 0.05 \\ -0.01 & -0.02 & 0 \end{pmatrix}$$

Для сходимости метода простых итераций достаточно, чтобы выполнялось условие ||A||<1 по какой-либо норме матрицы A:

$x^{(0)} := b \qquad x^{(k)} := b + A \cdot x^{(k-1)}$

$x =$

	0	1	2	3	4	5	6	7	8	9
0	2	1.92	1.907	1.907	1.907	1.907	1.907	1.907	1.907	1.907
1	3	3.19	3.188	3.189	3.189	3.189	3.189	3.189	3.189	3.189
2	5	4.92	4.917	4.917	4.917	4.917	4.917	4.917	4.917	4.917

$$\varepsilon := \frac{\left|x^{\langle 10 \rangle} - x^{\langle 9 \rangle}\right|}{\left|x^{\langle 9 \rangle}\right|}$$

$\varepsilon = 8.662 \cdot 10^{-4}$

Решите самостоятельно

Методом простых итераций решить СЛАУ:
$50x_1 + 25x_2 - 10x_3 = 40$
$12x_1 + 40x_2 + 10x_3 = 50$
$36x_1 - 15x_2 + 60x_3 = 120$

Решить эту систему уравнений любым из точных методов и сравнить результаты

Произвести qr - и lu-разложения матрицы коэффициентов при неизвестных. Убедиться, что матрица q – ортогональная.

Построить графики однополупериодного и двухполупериодного выпрямителей для функции

Y=2+3sin(x) (синусоида с постоянной составляющей)

Лабораторная работа № 6.

Тема: Решение переопределенных систем.

Очень часто при решении практических задач приходится решать систему алгебраических линейных уравнений вида **Ax=b**, когда число уравнений (т.е. строчек в матрице A) превышает число неизвестных в векторе - столбце x (или, что то же самое, число столбцов в матрице A). Это получается при достаточно большом наборе наблюдений с переменными параметрами. Когда уравнений и неизвестных немного, обычно убирают "лишние" уравнения - те, которые наиболее сильно отличаются от остальных. При решении больших систем этот метод не подходит - можно выбросить "не то" уравнение. При решения подобных систем используют метод наименьших квадратов для минимизации вектора невязки.

Вектор невязки обозначим **r = Ax - b.** Невязкой называется длина **|r|** вектора невязки **r**. Вектор x*, доставляющий минимум невязке |r| = |Ax - b|, называется нормальным обобщенным решением системы Ax = b и вычисляется по формуле:

$$\mathbf{x^* = (A^T * A)^{-1} A^T b.}$$

Пример

Найти нормальное обобщенное решение линейной системы Ax = b для двух различных вариантов правых частей уравнения:

$$A := \begin{pmatrix} 1 & 2 & 4 \\ 2 & 5 & 7 \\ 2 & 4 & 6 \end{pmatrix} \qquad b1 := \begin{pmatrix} 3 \\ 7 \\ 6 \end{pmatrix} \qquad b2 := \begin{pmatrix} 3 \\ 7 \\ 7 \end{pmatrix}$$

$AP := A^{-1} \cdot A$ $\quad bp1 := A^{-1} \cdot b1$

$bp2 := A^{-1} \cdot b2$ $\quad xn1 := AP^{-1} \cdot bp1$ $\quad xn2 := AP^{-1} \cdot bp2$

$$AP = \begin{pmatrix} 22 & 49 & 73 \\ 33 & 73 & 110 \\ 1.144 \times 10^{-14} & 1 & 1 \end{pmatrix} \qquad xn1 = \begin{pmatrix} \\ \\ \end{pmatrix} \qquad xn2 = \begin{pmatrix} 0.6 \\ -0.4 \\ \end{pmatrix}$$

Получено два решения: xn1 и xn2.

Вычислим невязку для обоих решений:

$r1 := |A \cdot xn1 - b1| \qquad r2 := |A \cdot xn2 - b2| \qquad r1 = 8.995 \times 10$

Как видно из расчетов, для системы Ax = =b1 невязка минимальна и равна нулю с точностью до погрешностей округления. Следовательно, эта система совместна.

Минимум невязки для системы Ax = b2 равен 0,447. Следовательно, эта система несовместна.

Проверим решение для первой системы

$r := |A \cdot x - b1|$

$$Ag = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Классическое решение x совместной системы Ax = b1 совпадает с нормальным обобщенным решением xn1, минимум невязки для точного решения равен нулю.

Решите самостоятельно

Найдите нормальное обобщенное решение линейной системы AX = b для двух вариантов правых частей b1 и b2.

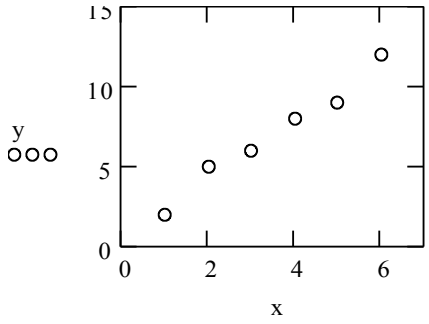
Найдите классическое решение системы, если оно существует, и сравните его с обобщенным

$$A := \begin{pmatrix} 1 & 1 & 1 \\ 1.5 & 1.333 & 1.25 \\ 2.5 & 2.333 & 2.25 \end{pmatrix} \qquad b1 := \begin{pmatrix} 2 \\ 2.833 \\ 4.833 \end{pmatrix} \qquad b2 := \begin{pmatrix} 3 \\ 3.833 \\ 5.833 \end{pmatrix}$$

2. ВЫПОЛНЕНИЕ РЕГРЕССИИ

При решении многих практических задач решение получается в виде таблицы. График полученного решения представляет собой "облако" точек, распределенных случайным образом, но подчиняющихся определенной закономерности. Задача регрессии заключается в получении параметров некоторой функции y=y(x) такими, чтобы функция приближала бы "облако" исходных точек с наименьшей среднеквадратической погрешностью.

$$x := \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \qquad y := \begin{pmatrix} 5 \\ 6 \\ 8 \\ 9 \\ 12 \end{pmatrix}$$



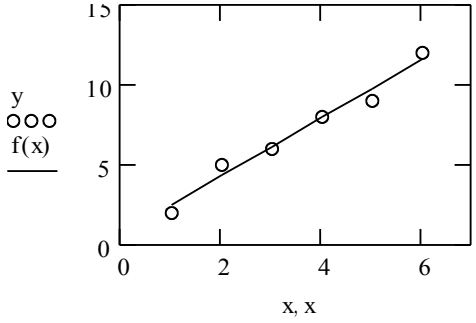
Очевидно, что переменная y зависит от переменной x (с ростом x переменная y также возрастает). Задача регресии- установить функциональную связь y=y(x).

Простейший вид уравнения регрессии - линейная регрессия. При этом график y(x) описывается уравнением прямой линии: **y = ax+b.**

Для проведения линейной регрессии в систему MathCad встроен ряд функций:

corr(VX,VY) - возвращает скаляр - коэффициент корреляции Пирсона;

intercept(VX,VY) - возвращает значение параметра b (свободного члена прямой регрессии)
slope(VX,VY) - возвращает значение параметра a (угловой коэффициент прямой регрессии)
VX,VY - исходные векторы



Как видно на рисунке, прямая регрессии проходит в "облаке" исходных точек с минимальным среднеквадратичным отклонением от них.
Вычислим невязку как сумму квадратов отклонений значений исходного вектора y и значений функции регрессии в точках x_i

$$RRR := \sum_i \left| y_i - f(x_i) \right|^2$$

В случае нелинейной зависимости между параметрами x и y можно использовать встроенную в MathCad функцию полиномиальной регрессии:
regress(VX,VY,n) - эта функция возвращает вектор **VS**, содержащий коэффициенты многочлена n-ной степени. Для вычисления коэффициентов полинома регрессии используется функция **submatrix**
Пример.

Полиномиальная регрессия

$$data := \begin{pmatrix} 2 & 3.5 \\ 3 & 4.4 \\ 4 & 15 \\ 5 & 20 \\ 6 & 18 \end{pmatrix}$$

x := data ... y := data ...

Степень полинома

z := (3 3 3 11.267 -16.042 6.863 -0.666)

coeffs := (11.267 -16.042 6.863 -0.666)

Таким образом, график полинома описывается уравнением:

$$y = 11{,}267 - 16{,}042x + 6{,}863x^2 - 0{,}666x^3$$

$$f1(x) := coeffs_0 + coeffs_1 \cdot x + coeffs_2 \cdot x^2 + coeffs_3 \cdot x^3$$

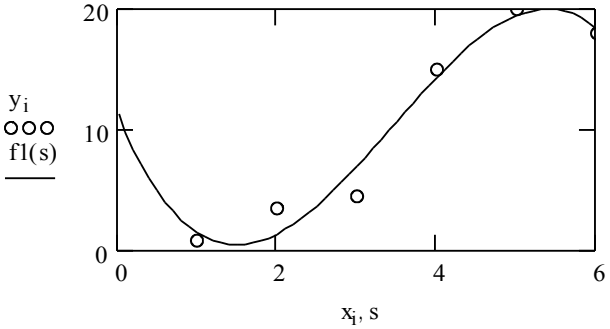


График полинома и исходные точки

На практике не рекомендуется делать степень аппроксимирующего полинома выше 4-6, так как сильно возрастают погрешности реализации регрессии.

Невязка равна:

$$RR := \sum_i \left| y_i - fl\left(x_i\right) \right|$$

Выполните самостоятельно:

Аппроксимировать функциональную зависимость между векторами vx и vy полиномом второго порядка. Вычислить невязку.

$$vx := \begin{pmatrix} 2 \\ 3 \\ 4 \\ 4.8 \\ 5.2 \end{pmatrix} \qquad vy := \begin{pmatrix} 7.5 \\ 9.4 \\ 16.2 \\ 26 \\ 32.1 \end{pmatrix}$$

Лабораторная работа № 7.

Тема: Решение нелинейных уравнений и систем.

Многие уравнения (например, трансцендентные) и системы из них не имеют аналитических решений. Однако они могут решаться численными методами с заданной погрешностью (но не более значения, заданного системной переменной TOL). Для простейших уравнений вида F(x) = 0 решение находится с помощью функции:

root(Выражение, Имя переменной)

Эта функция возвращает значение переменной, при котором выражение дает 0. Функция реализует вычисления итерационным методом, причем можно задать начальное значение переменной. Это особенно важно, если существует несколько решений. В этом случае выбор решения определяется выбором начального значения.

Пример 1.

Коэффициенты полинома

$F(x) := a3\,x^{-} + a2\,x^{-} + a1\,x + a0$

Задание полинома

Вычисление действительного корня

Вычисление других (возможно, комплексных) корней

$i := \sqrt{-1}$

Задание мнимой единицы

$x2 := root\left(\frac{}{x - x1}, x\right)$

$x3 := root\left(\frac{}{(x - x1)(x - x2)}, x\right)$

Для поиска всех корней полинома n-ной степени можно воспользоваться функцией:

polyroots(V) - эта функция возвращает вектор корней многочлена (полинома) степени n, коэффициенты которого находятся в векторе V, имеющем длину n+1 (так как существует "нулевой" коэффициент полинома).

0 1 2 3

Задание вектора V

$$polyroots(V) = \begin{pmatrix} \\ 1 - 3.464i \\ 4 \end{pmatrix}$$

вычисление корней полинома

Не рекомендуется пользоваться этой функцией, если степень полинома выше пятой-шестой, так как при этом сильно возрастает погрешность вычисления корней уравнения.

Решение систем нелинейных уравнений

При решении систем нелинейных уравнений используется специальный вычислительный блок, открываемый служебным словом - директивой **Given:**

Given
Уравнения
Ограничительные условия
Выражения с функциями Find и Minerr

Рекомендуется дополнить блок проверкой решения системы.

Find(v1,v2,...,vn) - возвращает значение одной или ряда переменных для точного решения;
Minerr(v1,v2,...,vn) - возвращает значение одной или ряда переменных для приближенного решения.
Примечание! Внутри блока вместо знака "присвоения" (двоеточия со знаком равенства :=) используется знак логического (символического) равенства (символ - жирный знак равенста =)
Между функциями **Find** и **Minerr** существуют принципиальные различия. Первая функция используется, когда решение реально существует (хотя и не является аналитическим). Вторая функция пытается найти максимальное приближение даже к несуществующему решению путем минимизации среднеквадратичной погрешности решения.

Пример 2. Найти корень уравнения $x^2=3$. Мы знаем, что решением будет значение плюс-минус корень квадратный из числа 3.

начальное приближение

$x^2 = 3$ Уравнение

Выражение с функцией Find

Найденное решение

Решение с применением функции Minerr также позволяет найти второе решение (обратите внимание, что начальное приближение выбрано равным –10).

$x^2 = 3$

Пример 3 Найти точки пересечения параболы $y=x^2$ и прямой **y=8+3x**.

диапазон построения графика

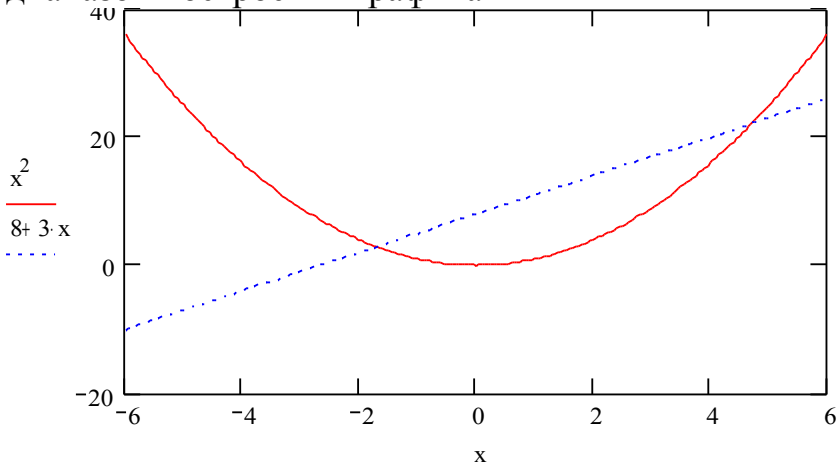


График показывает, что парабола и прямая пересекаются в двух точках: при x около -1.7 и при x около +4.8

Ищем первое решение при x=0 и y=0 и при ограничении x<0

$y = x^2$ $y = 8 + 3 \cdot x$

$\begin{pmatrix} y0 \end{pmatrix} := \text{Find}(x, y$

Найденное первое решение

$\begin{pmatrix} y0 \end{pmatrix} = \begin{pmatrix} 2.895 \end{pmatrix}$

Проверка решения (вычислением y):

$x0 = 2.895$

Найдем второе решение системы (подобно первому, но задав ограничение x>0)

$y = x^2$ $y = 8 + 3 \cdot x$

$\begin{pmatrix} y1 \end{pmatrix} := \text{Find}(x, y$

$\begin{pmatrix} y1 \end{pmatrix} = \begin{pmatrix} 22.105 \end{pmatrix}$ Найденное второе решение

Пример 4. Найти приближенное решение системы нелинейных уравнений $(x^2+1)^2+(y^2+1)^2=5.5$ и $x+y=0.95$

Начальные значения переменных

Начало вычислительного блока

$\left(x^2 + 1\right)^2 + \left(y^2 + 1\right)^2 = 5.5$

Заданная система уравнений

$x + y = 0.95$

$z = \begin{pmatrix} 1.056 \end{pmatrix}$ Найденное решение

Проверка решения:

$\left[\left(z_0\right)^2 + 1\right]^2 + \left[\left(z_1\right)^2 + 1\right]^2 = 5.5$

Решите самостоятельно:

1). Найти все корни уравнения $x^3=6$

2). Найти все корни уравнения $5x^3-14x^2+12x+9=0$

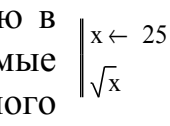
3). Построить графики и найти решение системы уравнений
$y=5x^2+2x-3$ и $x+y=1$

4). Найти приближенное решение системы нелинейных уравнений
$(x^2-4)^2-(y^2-1)^2=16$ и $x+y=0.5$

Лабораторная работа № 8.
Тема: Программирование.

Хотя MathCad является достаточно мощным средством, с помощью которого можно решить множество задач, существуют нестандартные задачи, не решаемые, не решаемые или решаемые малоэффективно стандартными средствами Маткада. Для решения подобных задач создана панель **Программирование** (в английской версии **Programming Toolbar**). Вызванная кнопкой панели управления, панель Программирование содержит 10 операторов, из которых строится программный блок. Следует отметить, что все операторы должны вводиться **только из палитры**, писать их **вручную не рекомендуется**. Панель содержит следующие элементы (операторы):

Add Line - создает и при необходимости расширяет жирную вертикальную линию, справа от которой в шаблонах задается запись программного блока. Вообще данный оператор имеет аналогию в языках программирования - так называемые **"операторные скобки"**, поэтому внутри операторного блока также можно проводить подобные жирные линии, выделяющие в единый блок несколько команд.



← - символ локального присваивания (в теле модуля). Внутри модуля **вместо** символа присваивания **:=** используется символ локального присваивания ←. Следует заметить, что присваивание действует только в теле модуля, то есть если какой-то переменной внутри модуля присвоено некоторое значение, то при выходе из модуля (после выполнения всех команд в нем) переменная теряет это значение и его необходимо присваивать заново. Кроме того, в теле модуля можно вводить новые локальные переменные, причем эти локальные переменные в разных модулям могут иметь одинаковые имена и никак не влияют друг на друга.

If - оператор условного выражения. Его также нельзя вводить с клавиатуры, так как этот оператор имеет другой формат, отличающийся от формата в основной программе. Формат оператора if внутри блока следующий: **(Оператор) If (Условие)**. Если выполняется **Условие**, то будет выполнен **Оператор.**

Рассмотрим пример программы, вычисляющий модуль (абсолютное значение) некоторого числа s. В блоке использован оператор **otherwise** - оператор иного выбора, использующийся совместно с оператором **if**.

$$m(s) := \begin{cases} s & \text{if } s > 0 \\ -s & \text{otherwise} \end{cases}$$

Другой пример - программа определяет четность и нечетность числа. Возвращается 0, если число четное и 1, если нечетное.

$$chet(a) := \begin{cases} 0 & \text{if } mod(a,2) = 0 \\ 1 & \text{otherwise} \end{cases}$$

Функция mod(s,n) возвращает остаток от деления числа s на число n

Если необходимо вычислить несколько значений функции, то необходимо поставить курсор на квадратик **Оператора** (слева от условия if) и "добавить линию" (Add Line)

```
q(a, b) := if a > b
             w ← a
             a ← b
             b ← w
           (a)
           (b)
```

Данный модуль записывает два числа в вектор-столбец, располагая числа по возрастанию

$q(4, 5) = \begin{pmatrix} 5 \end{pmatrix}$ $q(7, 3) = \begin{pmatrix} 7 \end{pmatrix}$

for - оператор задания цикла с фиксированным числом повторений

Переменная **Var** меняется с шагом +1 от значения Nmin до Nmax, при этом выполняется выражение, помещенное ниже в шаблоне.

```
sum(n) := s ← 0
          for i ∈ 1..n
            s ← s + i
```

Пример программы, вычисляющей сумму элементов от 1 до n

```
fakt(n) := g ← 1
           for i ∈ 1..n
             g ← g·i
```

Пример программы, вычисляющей факториал числа n

Следующий пример - определение максимальной координаты вектора y (max) и ее позиции (индекса) - ind:

```
kopos(y) := ind ← 0
            max ← y_0
            for i ∈ 1..last(y)
              if y_i > max
                max ← y_i
                ind ← i
            (max)
            (ind)
```

Начальное значение индекса

Начальное значение максимума (первый элемент вектора y)

Цикл по элементам вектора (оператор **last(y)** возвращает последний индекс вектора y)

Присваивание максимальному элементу max текущего значения вектора, а индексу ind- текущего индекса (в случае, когда выполняется условие if)

$g := \begin{pmatrix} 4 \\ -7 \\ 0 \\ 3 \\ 1 \end{pmatrix}$

Возвращение матрицы, содержащей два элемента максимум и индекс).

$kopos(g) = \begin{pmatrix} 1 \end{pmatrix}$

While - оператор, служащий для организации циклов, действующий до тех пор, пока выполняется некоторое условие. После оператора **while** записывается **условие**, а **выполняемое выражение** пишется ниже (в шаблоне).

Применение оператора while для вычисления факториала

```
Fakt(n) := | d ← 1
           | while n > 0
           |     d ← d·n
           |     n ← n − 1
           | d
```

```
Fact(n) := | f ← 1
           | while n ← n − 1
           |   f ← f·(n + 1)
           | f
```

break - оператор вызывает прерывание работы программы всякий раз, как он встречается. Обычно он используется совместно с оператором условного выражения **if** или операторами циклов **while** и **for,** обеспечивая переход в конец тела цикла.

continue - оператор продолжения используется для продолжения работы после прерывания программы. Он также обычно используется совместно с операторами циклов **while** и **for,** обеспечивая возврат после прерывания в начало цикла.

```
F(n) := | w ← n
        | while 1
        |     w ← w·(n − 1)
        |     n ← n − 1
        |     break if n = 1
        | w
```

Пример программы, вычисляющей факториал вводимого числа с использованием оператора **break**:

F(10) = 3.6288 × 10

Оператор -функция возврата **return** прерывает выполнение программы и возвращает значение своего операнда, стоящего следом за ним.

Например, в данном случае функция возвращает значение 0 при любом x<0 :

```
S(x) := | return 1 if x = 0
        | sin(x)/x  otherwise
```

Первый замечательный предел (при x=0) и значение функции в других точках.

Оператор обработки ошибок **on error** позволяет создавать конструкции обработчиков ошибок. Этот оператор задается в виде:

```
y1(x) := (1/x)·sin(x)
```

Выражение 1 **on error** *Выражение 2* Если при выполнении Выражения 1 возникает ошибка, то выполняется Выражение 2.

Функция **error(S)** -функция обработки ошибок, которая, будучи в программном модуле, возвращает окошко с надписью, хранящейся в символьной переменной **S** или в символьной константе (любой фразе в кавычках).

```
F(g) := | return "One"    if g = 1
        | return "Two"    if g = 2
        | error("No value")  otherwise
```

F(3) =

Примеры программ по обработке матриц

```
change(D) := | k ← cols(D) - 1
             | n ← rows(D) - 1
             | for i ∈ 0..n
             |   | s ← D[i,0]
             |   | D[i,0] ← D[i,k]
             |   | D[i,k] ← s
             | D
```

.Пусть требуется переставить местами первый и последний столбцы матрицы D (причем число столбцов и строк матрицы заранее неизвестно):

$$D := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix} \qquad change(D) = \begin{pmatrix} 4 & 2 & 3 & 1 \\ 4 & 2 & 3 & 1 \end{pmatrix}$$

```
Perestan(A, i, j) := | k ← cols(A) - 1
                     | for s ∈ 0..k
                     |   | w ← A[i,s]
                     |   | A[i,s] ← A[j,s]
                     |   | A[j,s] ← w
                     | A
```

Пусть требуется поменять местами i -тую и j - тую строки матрицы A

$$A := \begin{pmatrix} 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{pmatrix} \qquad Perestan(A, 0, 2) = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 \end{pmatrix}$$

В примере переставлены местами нулевая и вторая строки

Выполните самостоятельно.

1) Ввести матрицу произвольного размера. Создать программный модуль, подсчитывающий число положительных, отрицательных и нулевых элементов в каждой строке матрицы. (Замечание: требуется вывести матрицу, содержащую столько же строк, сколько в исходной, и всего три столбца).

2) Подсчитать сумму положительных и отрицательных элементов матрицы в каждом столбце.

3) Найти сумму и произведение элементов главной и побочной диагонали.(Замечание: требуется вывести всего четыре числа).

4). Задать произвольную матрицу (любых размеров). Для заданной матрицы создать две других (можно в двух разных программных модулях), являющихся зеркальным отражением исходной матрицы относительно: а) главной диагонали; б) вспомогательной (побочной) диагонали.

5). Ввести произвольное число w. Написать программный модуль, разбивающий данное число на простые множители. (Примечание: так как заранее неизвестны ни размер числа, ни количество простых множителей, рекомендуется воспользоваться оператором until)

Лабораторная работа № 9.

Тема: Обыкновенные дифференциальные уравнения.

Почти все функции MathCad предназначены для решения задачи Коши нормальных систем обыкновенных дифференциальных уравнений следующего вида:

y1' = f1(x,y1,y2,...yn)
y2' = f2(x,y1,y2,...yn)
.....
yn' = fn(x,y1,y2,...yn)
при заданных начальных условиях
$y1(x0) = y1_0$
$y2(x0) = y2_0$
.....
$yn(x0) = yn_0$

Численное решение этой задачи состоит в постороении таблицы приближенных значений y1,y2,...yn на отрезке [x0,xn] в точках $x_0,x_1,x_2,...,x_n$, которые называют узлами сетки.

В библиотеке встроенных функций MathCad есть функция **odesolve**, предназначенная для решения линейных дифференциальных уравнений. Функция odesolve решает задачу Коши с заданными начальными условиями
y(x0) = y0, y'(x0) = y01, ...
или простейшую краевую задачу, в которой заданы *n* граничных условий, определяющих значения искомой функции *y*(*x*) и ее производных на концах отрезка.

Перед обращением к функции **odesolve(x,b,step)** необходимо записать ключевое слово **Given**, а затем ввести уравнение и начальные либо граничные условия. Здесь **x** – имя переменной интегрирования (аргумента искомой функции), **b** – правый конец отрезка интегрирования, **step** – шаг, который используется при интегрировании уравнения методом Рунге-Кутта (этот аргумент необязателен, его можно опустить).

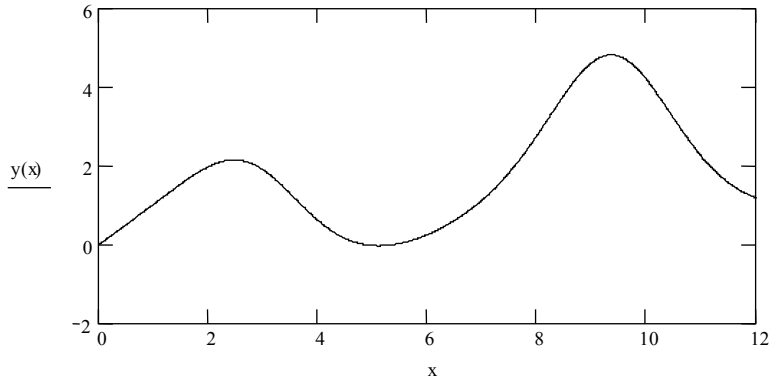
Пример 1. Найти с помощью функции odesolve на отрезке [0,4π] решение задачи Коши
y'' - y' sin(x) + y =x/(2π).
начальные условия: y(0)=0, y'(0)=1
Решение

Given

$$y''(x) - \sin(x)\cdot y'(x) + y(x) = \frac{x}{2\cdot\pi}$$

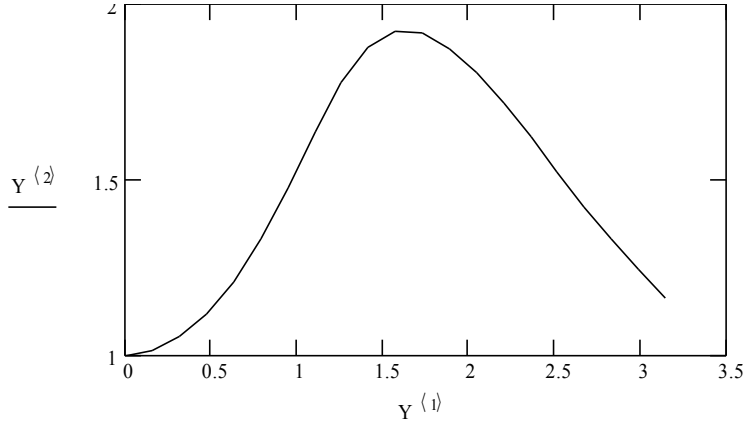
$y(0) = 0$ $y'(0) = 1$

y := odesolve(x,4,0.1)



Данную задачу можно также решить методом Рунге-Кутта при помощи функции **rkfixed(y,x1,x2,npoints,D) -** здесь **y** – вектор начальных значений $\mathbf{Y_0}$, **x1,x2** - начальная и конечная точки отрезка интегрирования системы ,**npoints** - число узлов на отрезке x1,x2, **D** – имя вектора-функции D(x,y), содержащей правые части уравнений, т.е. $\mathbf{D_i(x,y)=f_i\ (x,y1,y2,...,yn)}$

Пример 2 Решить уравнение **y' =sinxy** на отрезке $[0,\pi]$, удовлетворяющее начальным условиям y(0)=1. Решим задачу методом Рунге-Кутта с фиксированным шагом на сетке из 20 равноотстоящих узлов.



Обратите внимание, что перед обращением к функции rkfixed переменной y было присвоено начальное значение y_1, равное единице, а переменной D(x,y) - выражение для правой части уравнения, равное $\sin(x*y_1)$. Результаты вычислений присвоены матрице Y, которая в первом столбце содержит координаты 20 узлов равномерной сетки, а во втором – приближенные значения решения в этих узлах.

Решение задачи Коши для нормальной системы дифференциальных уравнений

Пример 3. Пусть требуется найти приближенное решение задачи Коши методом Рунге-Кутта

$y1' = -y2 + \sin x*y3$ $\qquad y1(0) = 1$

$y2' = -y1^2$ $\qquad y2(0) = 0$

y3' = -y3 - y1 y3(0) = 1

на отрезке [0,3] с фиксированным шагом на сетке из 30 равноотстоящих узлов.

$y1' = y2 + \sin(x \cdot y3)$

$y2' = -y1^2$

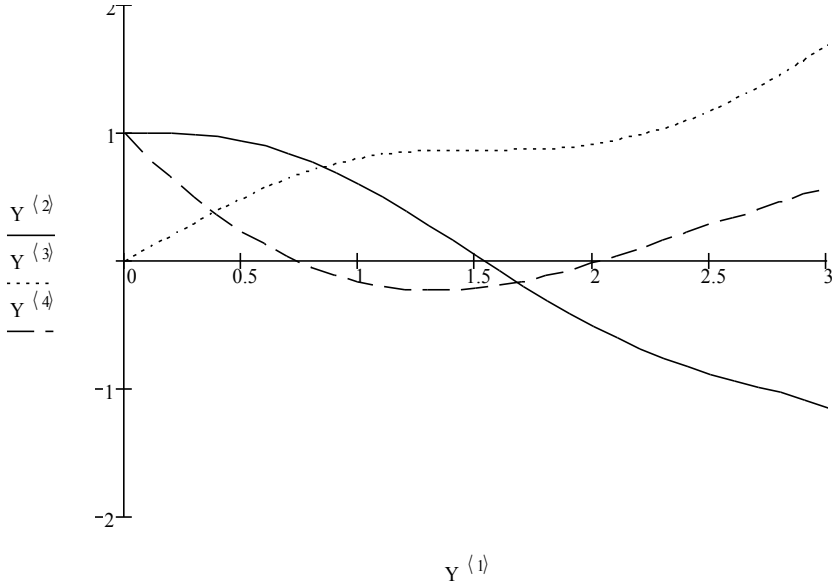
$y3' = -y3 - y1$

$y1(0) = 1$

$y2(0) = 0$

$y3(0) = 1$

$$D(x, y) := \begin{pmatrix} \\ (y_1)^2 \\ -y_3 - y_1 \end{pmatrix} \qquad y := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



Y =

	1	2	3	4
1	0	1	0	1
2	0.1	0.999	0.1	0.81
3	0.2	0.995	0.199	0.638
4	0.3	0.984	0.298	0.483
5	0.4	0.964	0.393	0.344
6	0.5	0.932	0.483	0.221
7	0.6	0.889	0.566	0.113
8	0.7	0.833	0.64	0.021
9	0.8	0.764	0.704	-0.057
10	0.9	0.683	0.757	-0.121
11	1	0.591	0.797	-0.17
12	1.1	0.49	0.827	-0.205
13	1.2	0.382	0.846	-0.227
14	1.3	0.268	0.857	-0.236
15	1.4	0.151	0.861	-0.234
16	1.5	0.032	0.862	-0.22

Выполните самостоятельно

1). Решите на отрезке **[0,5]** задачу Коши **y' = (y*lny)/sinx** с начальным условием **y(π/2)=e** по методу Рунге-Кутта с постоянным шагом (величину шага выберите самостоятельно). Изобразите графики решений , вычисленных с шагами **h, 2h и h/2.**

2). Решите задачу Коши для системы уравнений

y1' = y2*ln(x)

y2' = y1+y2^2

на отрезке **[1,4]** с постоянным шагом **h = 0.1** с заданными начальными условиями **y1(1) = - 2, y2(1) = - 1**

Лабораторная работа № 10.

Тема: Разложение функций в ряды.

В практике инженерных расчетов часто приходится прибегать к разложению функций в функциональные ряды, т.е. в ряды, составленные из функций определенного вида. Простейшим функциональным рядом является степенной ряд Тейлора:

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Точка x_0 называется центром разложения. При $x_0=0$ такой ряд называют рядом Маклорена:

$$\sum_{n=0} \frac{f^{(n)}(0)}{n!}x^n$$

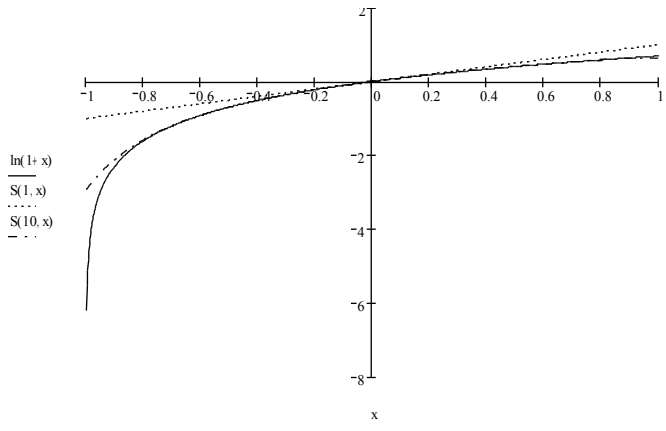
Функция f(x) может быть разложена в степенной ряд на интервале $(x_0+ R, x_0 - R)$, если существует степенной ряд, сходящийся к f(x) на данном интервале.

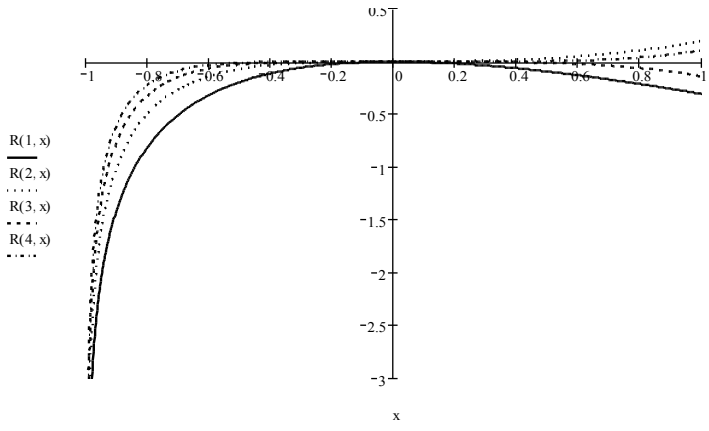
Пример 1

Рассмотрим разложение функции ln(1+x) в ряд Тейлора

$$c(n) := \frac{(-1)^{n+1}}{n} \qquad S(n,x) := \sum_{k=1} c(k)\cdot x^k \qquad \sum_{n=1} c(n)\cdot x^n \rightarrow \ln(1+x)$$

Для оценки точности вычислений вводится понятие остатка (остаточного члена) ряда R:





Пример 2

Разложить функцию 1\x в ряд Тейлора в окрестности точки x_0 = -2.Построить график функции и частичных сумм ее ряда Тейлора в окрестности этой точки.

t = x + 2

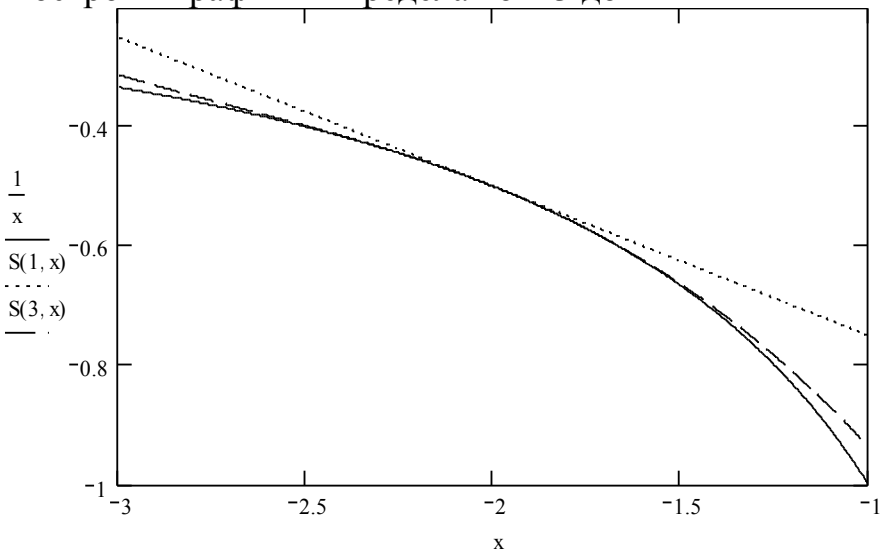
x = t - 2

$$c(n) := \frac{1}{2^{n+1}}$$

$$S(n, x) := \sum_{k=0} c(k)\cdot (x+2)^k$$

$$\sum_{k=0} c(k)\cdot (x+2)^k \to \frac{1}{x}$$

Построим графики в пределах от -3 до -1



Разложение функции в ряд Фурье

Ряд Фурье наиболее часто встречается в электротехнических расчетах. Каждой абсолютно интегрируемой на отрезке $[-\pi,\pi]$ функции f(x) можно поставить в соответствие ее тригонометрический ряд Фурье:

$$f(x) = \frac{a_0}{2} + \sum_{k=1} \left(a_k\cdot \cos kx + b_k\cdot \sin kx\right)$$

Коэффициенты ряда Фурье вычисляют по формулам Эйлера-Фурье:

$$a_k := \frac{1}{\pi}\cdot \int_{-\pi} f(x)\cdot \cos(k\cdot x)\, dx \quad , \quad b_k := \frac{1}{\pi}\cdot \int_{-\pi} f(x)\cdot \sin(k\cdot x)\, dx$$

Так как ряд Фурье есть бесконечная сумма, то иногда для расчетов удобнее пользоваться конечным числом членов ряда. Обозначим через $S_n(x)=\frac{a_0}{2} + \sum_{k=1}^{n} \left(a_k\cdot \cos kx + b_k\cdot \sin kx\right)$ n-ную частичную сумму ряда Фурье.

Пример 3. Построить график функции y= $\begin{matrix} 1, & |x| < \frac{\pi}{2} \\ 0, & |x| > \frac{\pi}{2} \end{matrix}$ и частичных сумм с различным числом членов.

$$f(x) := \begin{cases} 1 & \text{if } \frac{-\pi}{2} \le x \le \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}$$

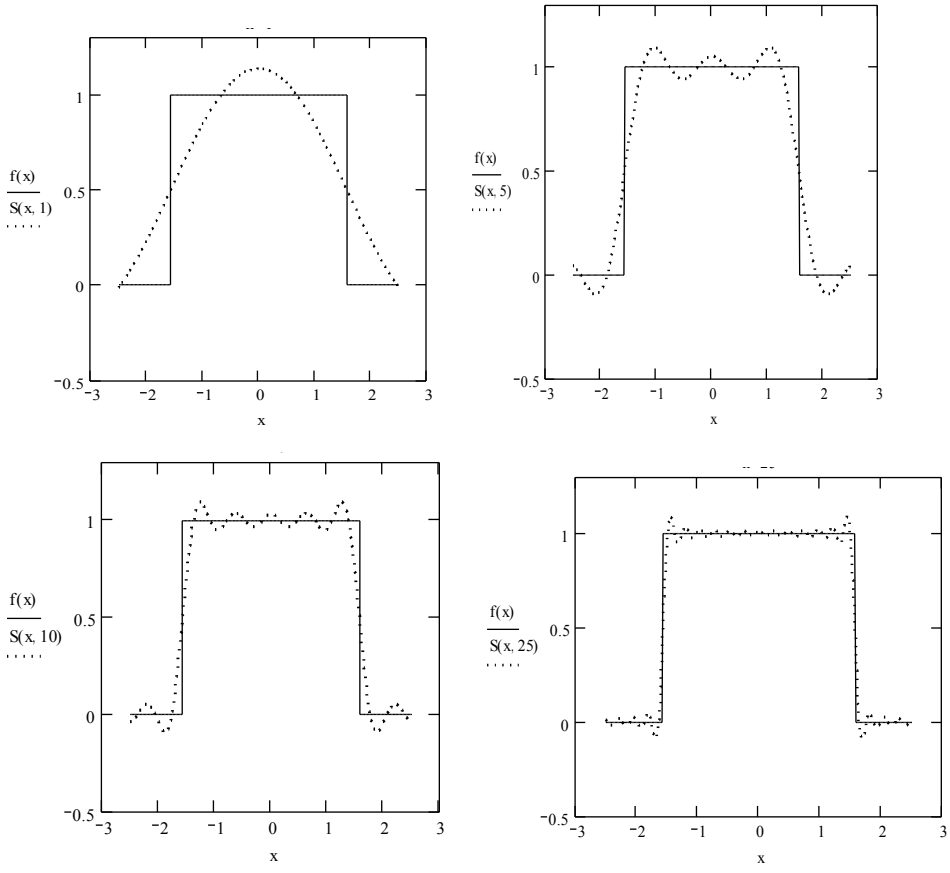
Вычисление коэффициентов ряда Фурье

$$a_k := \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) \cdot \cos(k \cdot x)\, dx \qquad b_k := \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) \cdot \sin(k \cdot x)\, dx$$

Вычисление частичной суммы ряда Фурье

$$S(x, n) := \frac{a_0}{2} + \sum_{k=1}^{n} \left(a_k \cdot \cos(k \cdot x) + b_k \cdot \sin(k \cdot x) \right)$$

Как видно из графиков, уже при n=25 цифровой сигнал достаточно хорошо описывается набором аналоговых сигналов (синусов и косинусов).



Выполните самостоятельно

1. Разложите в ряд Тейлора функцию $f(x)= e^x$ в окрестности точки x=0. Постройте графики для частичных сумм для n = 1,5,10,50.

2. Разложите в ряд Фурье функцию $f(x)= \begin{cases} -\frac{\pi + x}{2}, -\pi \le x \le 0 \\ \frac{\pi - x}{2}, 0 < x < \pi \end{cases}$

Постройте графики для частичных сумм для n = 1,2,5,10,25,50.

Лабораторная работа № 11

Тема: Решение дифференциальных уравнений при помощи преобразования Лапласа.

Одним из важнейших применений операционного исчисления является решение линейных дифференциальных уравнений с постоянными коэффициентами:

$$a_0 y^{(n)} + a_1 y^{(n-1)} + \ldots + a_n y = f(t) \qquad (1)$$

где $a_0, \ldots, a_n$ –заданные числа (коэффициенты уравнения),

f(t) – заданная функция,

y(t) – искомая (неизвестная) функция.

Требуется решить задачу Коши, т.е. найти решение этого уравнения, удовлетворяющее начальным условиям $y(0) = y_0$, $y'(0) = y'_0$, …, $y^{(n-1)}(0) = y_0^{(n-1)}$, где $y_0, y'_0, \ldots, y_0^{(n-1)}$ – заданные числа.

Перейдем от функций y(t) и f(t) к их изображениям Y(p) и F(p), используя свойство линейности преобразования Лапласа и теоремы о дифференцировании оригинала:

$$a_0 (p^n Y - p^{n-1} y_0 - p^{n-2} y'_0 - \ldots - y_0^{(n-1)}) + a_1 (p^{n-1} Y - p^{n-2} y_0 - p^{n-3} y'_0 - \ldots - y_0^{(n-2)}) + \ldots + a_n Y = F(p) \qquad (2)$$

Для перехода от функции f(t) к ее изображению F(p) используют таблицы перехода и основные теоремы, характеризующие свойства преобразования Лапласа.

№ п/п	Оригинал	Изображение	№ п/п	Оригинал	Изображение
1	1	$1/p$	11	$t \sin \omega t$	$2p\omega/(p^2+\omega^2)^2$
2	e^{at}	$1/(p-a)$	12	$t \cos \omega t$	$p^2 - \omega^2/(p^2+\omega^2)^2$
3	$\sin \omega t$	$\omega/(p^2+\omega^2)$	13	$t \operatorname{sh} \omega t$	$2p\omega/(p^2 - \omega^2)^2$
4	$\cos \omega t$	$p/(p^2+\omega^2)$	14	$t \operatorname{ch} \omega t$	$p^2+\omega^2/(p^2 - \omega^2)^2$
5	$\operatorname{sh} \omega t$	$\omega/(p^2-\omega^2)$	15	$(\sin \omega t)/t$	$\pi/2 - \operatorname{arctg}(p/\omega)$
6	$\operatorname{ch} \omega t$	$p/(p^2-\omega^2)$	16	$\alpha f(t)+\beta g(t)$	$\alpha F(p)+\beta G(p)$
7	$e^{at} \sin \omega t$	$\omega/((p-a)^2+\omega^2)$	17	$f(\lambda t)$	$(1/\lambda) F(p/\lambda)$
8	$e^{at} \cos \omega t$	$(p-a)/((p-a)^2+\omega^2)$	18	$f(t-\tau)$	$e^{-p\tau}F(p)$
9	t^n , n = 1,2,…	$n!/p^{n+1}$	19	$e^{-\lambda t} f(t)$	$F(p+\lambda)$
10	$t^n e^{at}$	$n!/(p-a)^{n+1}$	20	$f'(t)$	$P F(p) - f(0)$

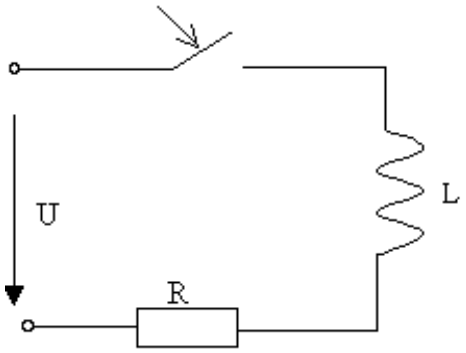
Уравнение (2) проще исходного дифференциального уравнения (1), так как оно является линейным алгебраическим уравнением относительно Y. Решая данное уравнение, находят Y(p), после чего выполняют обратное преобразование Лапласа – от найденного изображения Y(p) к оригиналу y(t).

Пример. Пусть требуется найти закон изменения электрического тока от времени для следующей электрической схемы:

Исходные данные для расчета: U = 100 В, R = 10 Ом, L = 0,5 Гн.

Дифференциальное уравнение, которое описывает процессы, происходящие в данной схеме, имеет следующий вид:

$$L \cdot \frac{di}{dt} + R \cdot i = U \qquad (3)$$



Перейдем от дифференциального уравнения относительно оригинала i(t) к уравнению относительно изображения I(p): **L(p I(p) – i(0)) + R I(p) = U/p**.

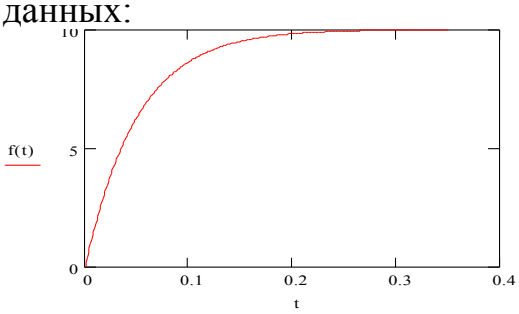
Выразив из этого уравнения I(p), получим решение уравнения (3) относительно его изображения (в переменных p):

$$I(p) = \frac{U + p \cdot L \cdot i(0)}{p \cdot (pL + R)} \qquad (4)$$

Зная изображение I(p), можно путем обратного преобразования Лапласа (функцией invlaplase) перейти обратно к оригиналу f(t) и построить график переходного процесса.

Перейдем к решению задачи в системе MathCad.

Ввод начальных данных:



Следует обратить внимание, что в Маткаде вместо переменной **p** используется переменная **s**.

Обратное преобразование Лапласа:

$$f(t) := \frac{0}{s \cdot (s \cdot L + R)}\ \text{invlaplace}, s \rightarrow 10. - 10. \cdot \exp(-20. \cdot t)$$

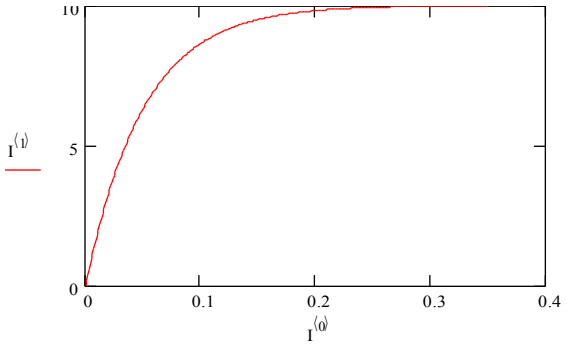
диапазон для построения графика

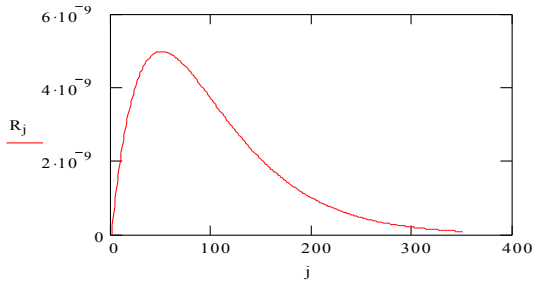
Проверка решения методом Рунге-Кутта:

$$L\frac{d}{dt}i + R \cdot i = U \qquad D(t, i) := \frac{U}{L} - \frac{0}{L}$$

Решение методом Рунге-Кутта

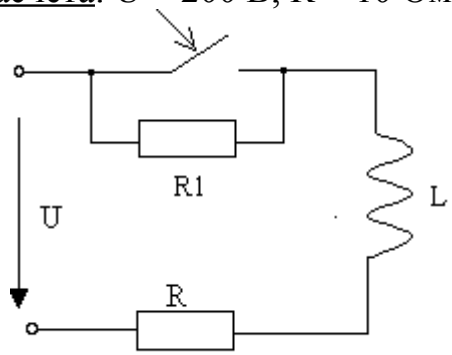
Невязка между методами:





Таким образом, максимальная невязка не превышает $5*10^{-9}$ А (при максимальном значении тока 10 А).

Выполнить самостоятельно. Построить графики переходных процессов для заданной схемы, в которой производится шунтирование сопротивления R1.
Исходные данные для расчета: U = 200 В, R = 10 Ом, R1 = 1,5 R, L = 0,5 Гн.



Требуется:

1) Используя преобразование Лапласа, решить заданную систему и найти закон изменения тока i(t) в цепи при включении шунта. Примечание: уравнения (3) и (4) справедливы и в данном случае. Необходимо лишь найти i(0). Также необходимо подобрать временной диапазон, в котором ток меняется от начальной величины до конечной.
2) Выполнить проверку правильности вычисления i(t), используя метод Рунге-Кутта. Вычислить невязку между методами и построить ее график, найти максимум невязки.
3) Исследовать влияние индуктивности L, построив три графика i(t) с различной величиной L: 0,5 Гн, 0,25 Гн и 0,1 Гн.
4) Исследовать влияние сопротивления R1, приняв три различных его значения: R1 = R, R1 = 2R, R1 = 0.1R. Построить три графика.

5. ФОНД ТЕСТОВЫХ И КОНТРОЛЬНЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ ПО ДИСЦИПЛИНЕ

Самостоятельная работа 1
Основы работы с MathCAD

Упражнение 1. Вычислить:

$\sqrt{100} =$ $|-10| =$ $10! =$.

Это и все остальные задания снабдить комментариями, используя команду **Вставка ⇒ Текстовая область**.

Упражнение 2. Определить переменные: $a := 3.4$, $b := 6.22$, $c \equiv 0.149$ (причем переменную c - глобально) и выражения:

$$Z := \frac{2ab + \sqrt[3]{c}}{\sqrt{\left(a^2 + b^{a+c}\right) \cdot c}} \qquad N := e^{\sin c} \cos\frac{a}{b}.$$

- Вычислить выражения.
- С помощью команды Формат⇒Результат⇒Формат чисел⇒Число знаков изменить точность отображения результатов вычисления *глобально*.

Упражнение 3. Вывести на экран значение *системной константы* π и установить максимальный формат ее отображения *локально*.

Упражнение 4. Выполнить следующие операции с комплексными числами:

$Z := -3 + 2i$ $|Z| =$ $Re(Z) =$ $Im(Z) =$ $arg(Z) =$

$\sqrt{Z} =$ $\sqrt{-5} =$ $2 \cdot Z =$ $Z1 := 1 + 2i$ $Z2 := 3 + 4i$

$Z1 + Z2 =$ $Z1 - Z2 =$ $Z1 \cdot Z2 =$ $Z1/Z2 =$

Упражнение 5. Выполнить следующие операции:

$i := 1 .. 10$ $\sum_i i =$ $\prod_i (i+1) =$ $\int_0^{0.4} x^2 \cdot \lg(x+2)dx =$ $\int_{0.8}^{1.2} \frac{\operatorname{ctg} 2x}{(\sin 2x)^2} dx =$

$x := 2$ $\frac{d}{dx} x^5 =$ $\frac{d}{dx} \sin(x) =$

Упражнение 6. Определить векторы *d, S* и *R* через дискретный аргумент *i*. Отобразить графически таблично заданные функции $S_i(d_i)$ и $R_i(d_i)$, используя команду **Вставка⇒График⇒X-Y Зависимость**. Чтобы оформить график, необходимо выполнить следующие команды:

i	d_i	S_i	R_i
0	0.5	3.3	2
1	1	5.9	3.9
2	1.5	7	4.5
3	2	6.3	3.7
4	2.5	4.2	1.2

- Щелкнуть левой клавишей мыши на графике, чтобы выделить его. Затем щелкнуть правой клавишей мыши, при этом появится контекстное меню в котором необходимо выбрать команду Формат (появится диалоговое окно “Formatting Currently Selected X-Y Plot”).
- Нанести линии сетки на график (Оси X-Y ⇒Вспом. линии) и отобразить легенду (След⇒Скрыть легенду)
- Отформатировать график так, чтобы в каждой узловой точке графика функции $S_i(d_i)$ стоял знак вида (**След⇒Символ⇒box**), а график функции $R_i(d_i)$ отобразить в виде гистограммы (**След⇒Тип⇒bar**).

Упражнение 7. Построить декартовы (**X-Y Зависимость**) и полярные (**Полярные Координаты**) графики следующих функций:

$X(\alpha) := \cos(\alpha) \cdot \sin(\alpha)$

$Y(\alpha) := 1.5\cos(\alpha)^2 - 1$

$P(\alpha) := \cos(\alpha).$

Для этого необходимо определить α как дискретный аргумент на интервале от 0 до $2\cdot\pi$ с шагом $\pi/30$.

Определить по графику **X-Y Зависимость** координаты любой из точек пересечения графиков $Y(\alpha)$ и $P(\alpha)$, для этого необходимо:

- Выделить график и выбрать из контекстного меню Масштаб (появится диалоговое окно “X-Y Zoom”) для увеличения части графика в области точки пересечения.
- На чертеже выделить пунктирным прямоугольником окрестность точки пересечения графиков Y(α) и P(α), которую нужно увеличить.
- Нажать кнопку Масштаб+, чтобы перерисовать график.
- Чтобы сделать это изображение постоянным, выбрать ОК.
- Выбрать из контекстного меню Трассировка (появится диалоговое окно “X-Y Trace”).
- Внутри чертежа нажать кнопку мыши и переместить указатель мыши на точку, чьи координаты нужно увидеть.
- Выбрать Copy X (или Copy Y), на свободном поле документа набрать Xper := (или Yper :=) и выбрать пункт меню Правка⇒Вставка.
- Вычислить значения функций $X(\alpha)$ и $Y(\alpha)$ при $\alpha:=\pi/2$.

Упражнение 8. Используя команду **Вставка⇒Матрица** создать матрицу Q размером 6×6, заполнить ее произвольно и отобразить графически с помощью команды **Вставка⇒График⇒Поверхности**.

Упражнение 9. Построить график поверхности (**Поверхности**) и карту линий уровня (**Контурный**) для функции двух переменных $X(t,\alpha) := t \cdot \cos(\alpha) \cdot \sin(\alpha)$, двумя способами:

- С помощью функции CreateMesh (сетка размером 40 × 40, диапазон изменения t от –5 до 5, α - от 0 до $2\cdot\pi$).
- Задав поверхность математически, для этого:
- Определить функцию X(t,α)
- Задать на осях переменных t и α по 41 точке

$$i:=0..40 \quad j:=0..40$$

для переменной t_i со значениями, изменяющимися от -5 до 5 с шагом 0.25 $t_i := -5 + 0.25 \cdot i$, а для переменной α_j - от 0 до $2\cdot\pi$ с шагом $\pi/20$ $\alpha_j := \pi/20 \cdot j$.

- Определить матрицу Mi j := X(ti,αj) и отобразить ее графически.
- С помощью команды Формат контекстного меню вызвать диалоговое окно “Формат 3-D графика” и изменить:
- характеристики просмотра (**Общее⇒Вид⇒Вращение, Наклон**),

- цвета и линии поверхности (**Внешний Вид⇒Свойства линии, Свойства заливки**),
- параметры осей (Оси),
- вид заголовка графика (**Название**).

Упражнение 10. Отобразить графически пересечение поверхностей $f1(x,y) := \frac{(x+y)^2}{10}$ и $f2(x,y) := 5\cdot\cos\left(\frac{x-y}{3}\right)$. Матрицы для построения поверхностей задать с помощью функции *CreateMesh*, значения факультативных параметров не указывать. Выполнить однотонную заливку для поверхностей, выбрав из контекстного меню команду **Формат**. Также из контекстного меню выбрать эффекты **Туман**, **Освещение, Перспектива**.

Упражнение 11. Используя переменную **FRAME** и команду **Вид ⇒ Анимация**, создать анимационные клипы с помощью данных приведенных в Таблице 1.

Таблица 1

Варианты упражнения 11

№ варианта	Переменные и функции	FRAME	Тип графика
1	$x := 0, 0.1 .. 30$ $f(x) := x + \text{FRAME}$	от 0 до 20	График Полярные Координаты
2	$i := 0 .. \text{FRAME} + 1$ $g_i := 0.5 \cdot i \cdot \cos(i)$ $h_i := i \cdot \sin(i)$ $k_i := 2 \cdot i$	от 0 до 50	3D точечный график границы на осях Min Max x - 50 50 y - 50 50 z 0 50 В метке для ввода матрицы укажите (g, h, k)
3	$i := 0 .. 20$ $j := 0 .. 20$ $f(x,y) := \sin(x^2 + y^2 + \text{FRAME})$ $x_i := -1.5 + 0.15 \cdot i$ $y_j := -1.5 + 0.15 \cdot j$ $M_{i,j} := f(x_i, y_j)$	от 0 до 50	График Поверхности В метке для ввода матрицы укажите M

Самостоятельная работа 2

Решение уравнений средствами Mathcad

Упражнение 1. Построить график функции $f(x)$ (Таблица 1) и приблизительно определить один из корней уравнения. Решить уравнение $f(x)= 0$ с точностью $\varepsilon = 10^{-4}$ с помощью встроенной функции Mathcad *root*;

Таблица 1

№ варианта	$f(x)$	№ варианта	$f(x)$
1	$e^{x-1} - x^3 - x$ $x \in [0, 1]$	**9**	$0.25x^3 + x - 2$ $x \in [0, 2]$
2	$x - \frac{1}{3 + \sin(3.6x)}$ $x \in [0, 1]$	**10**	$\arccos \frac{1 - x^2}{1 + x^2} - x$ $x \in [2, 3]$
3	$\arccos x - \sqrt{1 - 0.3x^3}$ $x \in [0, 1]$	**11**	$3x - 4\ln x - 5$ $x \in [2, 4]$
4	$\sqrt{1 - 0.4x^2} - \arcsin x$ $x \in [0, 1]$	**12**	$e^x - e^{-x} - 2$ $x \in [0, 1]$
5	$3x - 14 + e^x - e^{-x}$ $x \in [1, 3]$	**13**	$\sqrt{1 - x} - \operatorname{tg} x$ $x \in [0, 1]$
6	$\sqrt{2x^2 + 1.2 - \cos x} - 1$ $x \in [0, 1]$	**14**	$1 - x + \sin x - \ln(1 + x)$ $x \in [0, 2]$
7	$\cos\left(\frac{2}{x}\right) - 2\sin\left(\frac{1}{x}\right) + \frac{1}{x}$ $x \in [1, 2]$	**15**	$x^5 - x - 0{,}2$ $x \in [1, 2]$
8	$0.1x^2 - x\ln x$ $x \in [1, 2]$		

Упражнение 2. Для полинома $g(x)$ (Таблица 2) выполнить следующие действия:

1) с помощью команды **Символы ⇒ Коэффициенты полинома** создать вектор V, содержащий коэффициенты полинома;
2) решить уравнение g(x) = 0 с помощью функции polyroots;
3) решить уравнение символьно, используя команду **Символы ⇒ Переменные ⇒ Вычислить**.

Таблица 2

Варианты упражнения 2

№ варианта	$g(x)$	№ варианта	$g(x)$
1	$x^4 - 2x^3 + x^2 - 12x + 20$	**9**	$x^4 + x^3 - 17x^2 - 45x - 100$
2	$x^4 + 6x^3 + x^2 - 4x - 60$	**10**	$x^4 - 5x^3 + x^2 - 15x + 50$
3	$x^4 - 14x^2 - 40x - 75$	**11**	$x^4 - 4x^3 - 2x^2 - 20x + 25$
4	$x^4 - x^3 + x^2 - 11x + 10$	**12**	$x^4 + 5x^3 + 7x^2 + 7x - 20$
5	$x^4 - x^3 - 29x^2 - 71x - 140$	**13**	$x^4 - 7x^3 + 7x^2 - 5x + 100$
6	$x^4 + 7x^3 + 9x^2 + 13x - 30$	**14**	$x^4 + 10x^3 + 36x^2 + 70x + 75$
7	$x^4 + 3x^3 - 23x^2 - 55x - 150$	**15**	$x^4 + 9x^3 + 31x^2 + 59x + 60$
8	$x^4 - 6x^3 + 4x^2 + 10x + 75$		

Упражнение 3. Решить систему линейных уравнений (Таблица 3):

1) используя функцию *Find*;
2) матричным способом и используя функцию *lsolve*.

Таблица 3

Варианты упражнения 3

№ варианта	Система линейных уравнений	№ варианта	Система линейных уравнений
1	$\begin{cases} 2x_1 + x_2 + 2x_3 + 3x_4 = 8 \\ 3x_1 + 3x_3 = 6 \\ 2x_1 - x_2 + 3x_4 = 4 \\ x_1 + 2x_2 - x_3 + 2x_4 = 4 \end{cases}$	**9**	$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = -4 \\ x_1 - 3x_2 - 6x_4 = -7 \\ 2x_2 - x_3 + 2x_4 = 2 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = -2 \end{cases}$
2	$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 22 \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 17 \\ x_1 + x_2 + x_3 - x_4 = 8 \\ x_1 - 2x_3 - 3x_4 = -7 \end{cases}$	**10**	$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 26 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 34 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 26 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 26 \end{cases}$
3	$\begin{cases} 9x_1 + 10x_2 - 7x_3 - x_4 = 23 \\ 7x_1 - x_3 - 5x_4 = 37 \\ 5x_1 - 2x_3 + x_4 = 22 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 26 \end{cases}$	**11**	$\begin{cases} 2x_1 - 8x_2 - 3x_3 - 2x_4 = -18 \\ x_1 - 2x_2 + 3x_3 - 2x_4 = 28 \\ x_2 + x_3 + x_4 = 10 \\ 11x_2 + x_3 + 2x_4 = 21 \end{cases}$
4	$\begin{cases} 6x_1 - x_2 + 10x_3 - x_4 = 158 \\ 2x_1 + x_2 + 10x_3 + 7x_4 = 128 \\ 3x_1 - 2x_2 - 2x_3 - x_4 = 7 \\ x_1 - 12x_2 + 2x_3 - x_4 = 17 \end{cases}$	**12**	$\begin{cases} 2x_1 - x_2 + 4x_3 + x_4 = 66 \\ 2x_2 - 6x_3 + x_4 = -63 \\ 8x_1 - 3x_2 + 6x_3 - 5x_4 = 146 \\ 2x_1 - 7x_2 + 6x_3 - x_4 = 80 \end{cases}$
5	$\begin{cases} x_1 - 2x_2 + 6x_3 + x_4 = 88 \\ 5x_1 + 2x_3 - 3x_4 = 88 \\ 7x_1 - 3x_2 + 7x_3 + 2x_4 = 181 \\ 3x_1 - 7x_2 + 5x_3 + 2x_4 = 99 \end{cases}$	**13**	$\begin{cases} 2x_1 - 3x_3 - 2x_4 = -16 \\ 2x_1 - x_2 + 13x_3 + 4x_4 = 213 \\ 3x_1 + x_2 + 2x_3 + x_4 = 72 \\ x_1 - 12x_3 - 5x_4 = -159 \end{cases}$
6	$\begin{cases} x_1 - 2x_2 - 8x_4 = -7 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = -8 \\ x_1 + x_2 - 5x_3 + x_4 = -10 \\ 2x_1 - x_2 + 2x_4 = 7 \end{cases}$	**14**	$\begin{cases} 7x_1 + 7x_2 - 7x_3 - 2x_4 = 5 \\ 3x_1 + 4x_2 + 5x_3 + 8x_4 = 60 \\ 2x_1 + 2x_2 + 2x_3 + x_4 = 27 \\ 2x_1 - 2x_3 - x_4 = -1 \end{cases}$
7	$\begin{cases} 2x_1 + 2x_2 + 6x_3 + x_4 = 15 \\ -x_2 + 2x_3 + x_4 = 18 \\ 4x_1 - 3x_2 + x_3 - 5x_4 = 37 \\ 3x_1 - 5x_2 + x_3 - x_4 = 30 \end{cases}$	**15**	$\begin{cases} 6x_1 - 9x_2 + 5x_3 + x_4 = 124 \\ 7x_2 - 5x_3 - x_4 = -54 \\ 5x_1 - 5x_2 + 2x_3 + 4x_4 = 83 \\ 3x_1 - 9x_2 + x_3 + 6x_4 = 45 \end{cases}$
8	$\begin{cases} 4x_1 - 5x_2 + 7x_3 + 5x_4 = 165 \\ 2x_1 + x_2 - 3x_3 - x_4 = -15 \\ 9x_1 + 4x_3 - x_4 = 194 \\ x_1 - x_2 - 2x_3 - 3x_4 = -19 \end{cases}$		

Упражнение 4. Преобразовать нелинейные уравнения системы из Таблицы 4 к виду $f_1(x) = y$ и $f_2(y) = x$. Построить их графики и определить начальное приближение решения. Решить систему нелинейных уравнений с помощью функции *Minerr*.

Таблица 4

Варианты упражнения 4

№ варианта	Система нелинейных уравнений	№ варианта	Система нелинейных уравнений
1	$\begin{cases} \sin x + 2y = 2, \\ \cos(y - 1) + x = 0{,}7. \end{cases}$	**9**	$\begin{cases} \sin y + x = -0{,}4, \\ 2y - \cos(x + 1) = 0. \end{cases}$
2	$\begin{cases} \sin(x + 0{,}5) - y = 1, \\ \cos(y - 2) + x = 0. \end{cases}$	**10**	$\begin{cases} \sin(x + 2) - y = 1{,}5, \\ \cos(y - 2) + x = 0{,}5. \end{cases}$

3	$\begin{cases}\cos x + y = 1{,}5,\\ 2x - \sin(y - 0{,}5) = 1.\end{cases}$	**11**	$\begin{cases}\cos(x + 0{,}5) - y = 2,\\ \sin y - 2x = 1.\end{cases}$
4	$\begin{cases}\cos(x + 0{,}5) + y = 0{,}8,\\ \sin y - 2x = 1{,}6.\end{cases}$	**12**	$\begin{cases}\cos(x - 2) + y = 0,\\ \sin(y + 0{,}5) - x = 1.\end{cases}$
5	$\begin{cases}\sin(x - 1) = 1.3 - y,\\ x - \sin(y + 1) = 0.8.\end{cases}$	**13**	$\begin{cases}\cos(x + 0{,}5) + y = 1,\\ \sin(y + 0{,}5) - x = 1.\end{cases}$
6	$\begin{cases}\cos(x + 0{,}5) + y = 1,\\ \sin y - 2x = 2.\end{cases}$	**14**	$\begin{cases}\sin(x) - 2y = 1,\\ \cos(y + 0{,}5) - x = 2.\end{cases}$
7	$\begin{cases}-\sin(x + 1) + y = 0{,}8,\\ \sin(y - 1) + x = 1{,}3.\end{cases}$	**15**	$\begin{cases}2y - \sin(x - 0{,}5) = 1,\\ \cos(y) + x = 1{,}5.\end{cases}$
8	$\begin{cases}\sin(x) - 2y = 1,\\ \sin(y - 1) + x = 1{,}3.\end{cases}$		

Упражнение 5. Символьно решить системы уравнений:

$$\begin{cases}3x + 4\pi y = a,\\ 2x + y = b.\end{cases} \qquad \begin{cases}2y - \pi z = a,\\ \pi z - z = b,\\ 3y + x = c.\end{cases}$$

Самостоятельная работа 3
Символьные вычисления

Упражнение 1. Используя операцию **Символы ⇒ Расчеты ⇒ С плавающей запятой…**, представьте:

1) число π в 7 позициях;
2) число 12, 345667 в 3 позициях.

Упражнение 2. Выведите следующие числа в комплексной форме, используя операцию **Расчеты ⇒ Комплексные** меню **Символы**:

1) $\sqrt{-7}$;
2) tg $(a\sqrt{-3})$;
3) $e^{1+\frac{\pi}{4}i}$;
4) для выражения 3) последовательно выполните операции Расчеты ⇒ Комплексные и **Упростить** меню **Символы.**

Упражнение 3. Для полинома $g(x)$ (см. Таблица 1) выполнить следующие действия:

1) разложить на множители, используя операцию **Символы ⇒ Фактор;**
2) подставьте выражение x = y + z в g(x), используя операцию Символы ⇒ Переменные ⇒ Замена (предварительно скопировав подставляемое выражение в буфер обмена, выделив его и нажав комбинацию клавиш Ctrl + C);
3) используя операцию Символы ⇒ Расширить, разложите по степеням выражение, полученное в 2);
4) используя операцию Символы ⇒ Подобные, сверните выражение, полученное в 3), по переменной z.

Варианты упражнения 3

№ варианта	g(x)	№ варианта	$g(x)$
1	$x^4 - 2x^3 + x^2 - 12x + 20$	9	$x^4 + x^3 - 17x^2 - 45x - 100$
2	$x^4 + 6x^3 + x^2 - 4x - 60$	10	$x^4 - 5x^3 + x^2 - 15x + 50$
3	$x^4 - 14x^2 - 40x - 75$	11	$x^4 - 4x^3 - 2x^2 - 20x + 25$
4	$x^4 - x^3 + x^2 - 11x + 10$	12	$x^4 + 5x^3 + 7x^2 + 7x - 20$
5	$x^4 - x^3 - 29x^2 - 71x - 140$	13	$x^4 - 7x^3 + 7x^2 - 5x + 100$
6	$x^4 + 7x^3 + 9x^2 + 13x - 30$	14	$x^4 + 10x^3 + 36x^2 + 70x + 75$
7	$x^4 + 3x^3 - 23x^2 - 55x - 150$	15	$x^4 + 9x^3 + 31x^2 + 59x + 60$
8	$x^4 - 6x^3 + 4x^2 + 10x + 75$		

Упражнение 4. Разложите выражения на элементарные дроби используя операцию **Символы ⇒ Переменные ⇒ Преобразование в частичные доли:**

1) $\frac{6x^2 - x + 1}{x^3 - x}$;
2) $\frac{3x^2 - 2}{(x^2 + x + 1)(x + 1)}$;
3) $\frac{x + 1}{x(x - 1)^3}$;
4) $\frac{5x^2 - 4x + 16}{(x^2 - x + 1)^2(x - 3)}$.

Упражнение 5. Разложите выражения в ряд с заданной точностью, используя операцию **Символы ⇒ Переменные ⇒ Разложить на составляющие:**

1) $\ln(1+x)$, $x_0 = 0$, порядок разложения 6;
2) $\sin(x)^2$, $x_0 = 0$, порядок разложения 6.

Упражнение 6. Найти первообразную аналитически заданной функции $f(x)$ (Таблица 4), используя операцию **Символы ⇒ Переменные ⇒ Интеграция**.

Упражнение 7. Определить символьное значение первой и второй производных $f(x)$ (Таблица 4), используя команду **Символы ⇒ Переменные ⇒ Дифференциалы**.

Таблица 4

Варианты упражнений 6 и 7

№ варианта	$f(x)$	№ варианта	$f(x)$	№ варианта	$f(x)$
1	$1/(\mathrm{tg}2x+1)$	6	$x^2 \cdot \mathrm{arctg}(x/3)$	11	$(2x+3)\sin x$
2	$\cos x/(2x+5)$	7	$e^{2x}\sin 3x$	12	$\cos 3x/(1-\cos 3x)^2$
3	$1/(x\sqrt{x^3+4})$	8	$\mathrm{ctg}2x/(\sin 2x)^2$	13	$1/(1+x+x^2)$
4	$\sin x/(1+\sin x)$	9	$(x+1)\sin x$	14	$(1+x)/(2+x)$
5	$x^2 \cdot \lg(x+2)$	10	$5x + x\lg x$	15	$\sqrt{1+e^{-x}}$

Упражнение 8.

1) Транспонируйте матрицу M

$$\begin{pmatrix} 1 & a & b \\ x & 2 & c \\ x^2 & 3 & d \end{pmatrix}$$

с помощью операции **Символы ⇒ Матрицы ⇒ Транспонирование.**

2) Инвертируйте матрицу

$$\begin{pmatrix} 1 & y \\ x & 2 \end{pmatrix}$$

с помощью операции Символы ⇒ Матрицы ⇒ Инвертирование.

3) Вычислите определитель матрицы M

$$\begin{pmatrix} 1 & a & b \\ x & 2 & c \\ x^2 & 3 & d \end{pmatrix}$$

с помощью операции **Символы ⇒ Матрицы ⇒ Определитель**.

Упражнение 8. Вычислите пределы:

1) $\lim_{x\to 1} \dfrac{x^2+2\cdot x+5}{x^2+1}$

2) $\lim_{x\to \frac{\pi}{2}} (2\cdot\sin(x)-\cos(x)+\mathrm{ctg}(x))$

3) $\lim_{h\to 0} \dfrac{(x+h)^3-x^3}{h}$

4) $\lim_{x\to\infty} \dfrac{\sqrt{x^2-3}}{\sqrt[3]{x^3}+1} \to 1$

5) $\lim_{x\to+\infty} x\cdot\left(\sqrt{x^2+1}-x\right)$

6) $\lim_{x\to-\infty} x\cdot\left(\sqrt{x^2+1}-x\right) \to -\infty$

7) $\lim_{x \to 0^+} (1 + x)^{\frac{1}{x}}$ 8) $\lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$

Упражнение 9. Задайте операторы пользователя:

1) Для пересчета единиц электрической энергии (кВт·ч в Дж, эВ в Дж) если известно, что

1 кВт·ч = $3{,}6 \cdot 10^6$ Дж;

1 эВ = $1{,}602 \cdot 10^{-19}$ Дж.

2) Для пересчета единиц магнитной индукции (Вб/см^2 в Т, Гс в Т) если известно, что

1 Вб/см^2 = $1 \cdot 10^4$ Т;

1 Гс = $1 \cdot 10^{-4}$ Т.

3) Для пересчета единиц мощности (эрг/с в Вт, кгс·м/с в Вт) если известно, что

1 эрг/с = $1 \cdot 10^{-7}$ Вт;

1 кгс·м/с = 9,80665 Вт.

Самостоятельная работа 4

Построение графиков и символьные вычисления

Задание 1. Построить графики функций. Найти точки разрыва заданных функций и определить их тип.

Варианты:

1. $\sqrt[3]{x-1}\cdot \operatorname{arctg}\dfrac{1}{x-1}$;
2. $\ln\left(1+(x+1)^2\cdot \sin^2\dfrac{1}{x+1}\right)$;
3. $\dfrac{2}{1+\exp(-1/x)}$;
4. $\sin(x+2)\cdot \sin\dfrac{1}{x+2}$;
5. $\ln\left(2+x\cdot\cos\dfrac{1}{2}\right)$;
6. $\operatorname{tg}\sqrt[3]{x-1}\cdot \operatorname{arctg}\dfrac{1}{x-1}$;
7. $(e^{x-1}-1)\cdot \sin\dfrac{1}{x-1}$.

Задание 2. Изобразить кривую, заданную в полярных координатах.

Варианты:

1. $\dfrac{2}{\cos(\varphi/2)}$;
2. $-2\cdot\operatorname{ctg}\varphi$;
3. $2\cdot\cos\varphi+2$;
4. $2\cdot\operatorname{ctg}\varphi$;
5. $2\cdot(1-\cos\varphi)$;
6. $\dfrac{2}{\sin\varphi}+2$;
7. $5\cdot\sin\dfrac{4\varphi}{3}$.

Задание 3. Построить графики функции и найти наибольшее и наименьшее ее значение на заданном отрезке.

Варианты:

1. $y=x^2+\dfrac{16}{x}-16$ на отрезке [1, 4].
2. $y=\sqrt[3]{2x^2(x-6)}$ на отрезке [-2, 4].
3. $y=-\dfrac{2(x^2+3)}{x^2+2x+5}$ на отрезке [-5, 1].
4. $y=\sqrt[3]{2x^2(x-3)}$ на отрезке [-1, 6].
5. $y=\dfrac{x+1}{x}$ на отрезке [0.1, 1.5].
6. $y=e^x-x^2+12$ на отрезке [-5, -3].
7. $y=\sin x+\cos x-1.78\cdot e^{x-3}$ на отрезке [1, 2].

Задание 4. Изобразить график поверхности, а также линии уровня функции $y=x^k\cdot y^m\cdot\exp\left(-\dfrac{x^2}{A}-\dfrac{y^2}{B}\right)$ в квадрате $-a<x<a$, $-a<y<a$. Указать приближенно координаты локальных экстремумов и седловых точек, если они есть.

Изучить возможности изменения параметров рисунка.

Порядок выполнения задания:

1. Определить дискретные аргументы i и j.
2. Определить в плоскости значения аргументов x_i и y_i.
3. Определить функцию двух переменных x и y: $f(x, y)$.
4. Заполнить матрицу M значениями функции $f(x_i, y_i)$.

5. Построить график поверхности.
6. Рассмотреть все возможности изменения графика с помощью диалогового окна форматирования (вращение, наклон, окрашивание и т.д.).
7. Построить график из линий равного уровня, или контурный график, активизируя опцию Contour Plot (Линии уровня).

Варианты (вариант определяет конкретный набор параметров заданной функции):

№	*k*	*m*	A	B	*a*
1	1	1	3	4	4
2	2	1	3	4	4
3	1	2	3	4	4
4	2	2	3	4	4
5	2	2	3	2	4
6	2	2	2	2	2
7	2	2	4	2	4

Задание 5. Вычислить таблицу значений функции, ее первой и второй производных для набора значений аргумента. Построить графики.

Порядок выполнения задания:

1. Определить исходную функцию $f(x)$.
2. Определить функцию, являющуюся первой производной $g(x) = f'(x)$.
3. Определить функцию, являющуюся второй производной $h(x) = f''(x)$.
4. Задать диапазон изменения аргумента.
5. Вывести столбцы значений $f(x)$, $g(x)$, $h(x)$.
6. Построить графики.

Варианты:

1. $f(x) = x^4 - 2.2\,x^3 + 0.5\,x^2 - 7x - 3.4$;
2. $f(x) = x^4 - 3.2\,x^3 + 1.5\,x^2 - 7x - 5.4$;
3. $f(x) = x^4 - 5.2\,x^3 + 2.5\,x^2 - 7x - 2.4$;
4. $f(x) = x^4 - 4.2\,x^3 + 3.5\,x^2 - 7x - 7.4$;
5. $f(x) = x^4 - 2.2\,x^3 + 7.5\,x^2 - 7x - 3.9$;
6. $f(x) = x^4 - 2.9\,x^3 + 6.5\,x^2 - 7x - 5.4$;
7. $f(x) = x^4 - 3.2\,x^3 + 9.5\,x^2 - 7x - 7.5$.

Задание 6. Вычислить сумму (или произведение) как функцию от числа суммируемых членов n для $n = 1, 2, \ldots, 10$ при заданном значении аргумента x. Построить график зависимости, на котором обозначить точное значение. Точное значение для каждого варианта задания приведено в квадратных скобках в виде выражения, представляющего данную сумму, или произведение как функцию от x либо в виде табличного значения (при $x = 1$), взятого из справочника.

Варианты:

1. $S = \sum_{k=0}^{n} \frac{x^{2k+1}}{(2k+1)!}$, [sh (x)];

2. $S = \sum_{k=0}^{n} \frac{(-1)^k x^{2k}}{(2k)!}$, [cos (*x*)];

3. $S = \sum_{k=1}^{n} \frac{x^k}{kk!}$, [S (1) = 1.3179];

4. $P = \prod_{k=0}^{n} \left(1 - \frac{4x^{2k}}{(2k+1)^2 \pi^2}\right)$, [cos (*x*)];

5. $P = \frac{(\pi + 2x)^2}{8} \prod_{k=1}^{n} \left(1 - \left(\frac{\pi - 2x}{2k\pi}\right)^2\right)$, [1 + sin (*x*)];

6. $S = \sum_{k=1}^{n} \frac{(-1)^{k+1} x^{2k}}{(2k)!(2k)!}$, [S (1) = 0.2398];

7. $P = x \cdot \prod_{k=1}^{n} \left(1 - \frac{x^2}{k^2 \pi^2}\right)$, [sin (*x*)].

Задание 7.

Вычислить неопределенный интеграл $\int f(x)\, \mathrm{d}\, x$, проверить правильность вычислений, взяв производную, и построить графики семейства первообразных.

Порядок выполнения задания:

1. Задать подынтегральную функцию как функцию аргумента *x*.
2. Найти первообразную, используя панель Symbolic.
3. Определить первообразную как функцию переменной.
4. Найти производную первообразной, используя панель Symbolic.
5. Построить на одном графике изображения нескольких первообразных.
6. Найти определенный интеграл от подынтегральной функции на отрезке [*a*, *b*].
7. Найти разность значений первообразной F(*b*) – F(*a*).
8. Сравнить полученные результаты.

Варианты.

1. $\frac{1}{\sin^2 x \cdot (1 - \cos x)}$; 2. $\frac{\cos x - \sin x}{(1 + \sin x)^2}$; 3. $\frac{\cos x}{1 + \sin x - \cos x}$; 4. $\frac{\sin x}{2 + \sin x}$;

5. $\frac{1}{\sin x \cdot (1 + \sin x)}$; 6. $\frac{\sin x}{2 + \sin x}$; 7. $\frac{\cos x}{2 + \sin x}$.

Задание 8. Найти асимптоты и построить графики функций.

Порядок выполнения задания:

1. Определите функцию $f(x)$ и постройте ее график.
2. Найдите точку пересечения с осью ординат, вычислив $f(0)$.
3. Найдите точку пересечения с осью абсцисс, решив уравнение $f(x) = 0$.
4. Найдите точки разрыва функции, вычислите соответствующие односторонние пределы, запишите уравнение вертикальных асимптот.

5. Вычислите пределы $k = \lim\limits_{x \to \infty} \dfrac{f(x)}{x}$ и $b = \lim\limits_{x \to \infty} f(x) - kx$; изобразите прямую $y = kx + b$.

6. Покажите, что равен нулю предел на бесконечности расстояния d от кривой до наклонной асимптоты ($d \le |f(x) - (kx + b)|$).

Варианты.

1. $y = \dfrac{4x^2 + 9}{4x + 8}$; 2. $y = \dfrac{2x^2 - 1}{\sqrt{x^2 - 2}}$; 3. $y = \dfrac{x^3 - 5x}{5 - 3x^2}$;

4. $y = \dfrac{x + 16}{\sqrt{9x^2 - 8}}$; 5. $y = \dfrac{-8 - x^2}{\sqrt{x^2 - 4}}$.

Задание 9. Символьно вычислить.

Варианты.

1. $\lim\limits_{x \to 0} \left(\dfrac{\sin(2x)}{x} \right)^{1 + x}$.

2. $\lim\limits_{x \to 1} \dfrac{1 + \cos(\pi x)}{\operatorname{tg}^2(\pi x)}$.

3. Разложить в ряд Тейлора по степеням x: $x^2 \cdot \sqrt{4 - 3x}$; $\ln(1 + x - 12x^2)$.

4. Разложить $\dfrac{x}{\sqrt{1 + x}}$ в степенной ряд по степеням $\dfrac{x}{1 + x}$.

5. Разложить в ряд Фурье: $\sin^4 x$; $\sum\limits_{i=0}^{n} \left(\alpha_i \cos(ix) + \beta_i \sin(ix) \right)$.

6. Разложить $x^x - 1$ по целым положительным степеням бинома $x - 1$ до члена $(x - 1)^3$.

7. Разложить $\ln \dfrac{1}{2 + 2x + x^2}$ по степеням бинома $x + 1$.

Самостоятельная работа 5

Действия с векторами и матрицами

Задание 1. Доказать, что матрица P идемпотентна. Вычислить ее определитель. Показать, что матрица I = 2P – E инволютна. Вычислить ее определитель и обратную матрицу.

Для справки. Матрица P называется идемпотентной, если $P^2 = P$. Матрица I называется инволютной, если $I^2 = E$.

Порядок выполнения задания:

1.Ввести матрицу P.

2.Вычислить P2 и P2 – P.

3.Сделать вывод об идемпотентности матрицы P, используя команду Text Region (Текстовая область) из меню Insert (Вставка).

4.Вычислить det P и P-1.

5.Ввести единичную матрицу E той же размерности, что и матрица P.

6.Вычислить матрицу I = 2P – E.

7.Вычислить матрицу I2.

8.Сделать вывод об инволютности матрицы I, используя команду Text Region из меню Insert.

9.Вычислить det I и I^{-1}.

Варианты:

1.	$P := \begin{pmatrix} 0.961 & 0.134 & 0.140 \\ 0.134 & 0.539 & -0.480 \\ 0.140 & -0.480 & 0.500 \end{pmatrix}$		4.	$P := \begin{pmatrix} 0.646 & -0.227 & -0.421 \\ -0.227 & 0.854 & -0.270 \\ -0.421 & -0.270 & 0.500 \end{pmatrix}$	
2.	$P := \begin{pmatrix} 0.789 & -0.247 & -0.325 \\ -0.247 & 0.711 & 0.380 \\ -0.325 & 0.380 & 0.500 \end{pmatrix}$		5.	$P := \begin{pmatrix} 0.990 & 0.070 & -0.071 \\ 0.070 & 0.510 & 0.495 \\ -0.071 & 0.495 & 0.500 \end{pmatrix}$	
3.	$P := \begin{pmatrix} 0.959 & -0.138 & -0.144 \\ -0.138 & 0.541 & -0.479 \\ 0.144 & 0.479 & 0.500 \end{pmatrix}$		6.	$P := \begin{pmatrix} 0.852 & -0.228 & 0.272 \\ -0.228 & 0.648 & 0.420 \\ 0.272 & 0.420 & 0.500 \end{pmatrix}$	

Задание 2. Докажите, что векторы f1, f2, f3, f4 образуют базис в пространстве L_4, найдите координаты вектора $x=(1\ 1\ 1\ 1)^T$ в этом базисе.

Порядок выполнения задания:

1. Введите матрицу C со столбцами f1, f2, f3, f4.
2. Покажите, что определитель матрицы C отличен от нуля.
3. Вычислите матрицу, обратную к матрице C.
4. Запишите выражение и вычислите координаты вектора в базисе f1, f2, f3, f4.
5. Проверьте вычисления, выполнив обратный переход.

Варианты:

1.	$f_1 := \begin{pmatrix} 0.500 \\ 0.667 \\ 0.400 \\ 0.200 \end{pmatrix}$	$f_2 := \begin{pmatrix} 0.667 \\ 0.500 \\ 0.333 \\ 0.286 \end{pmatrix}$	$f_3 := \begin{pmatrix} 1.500 \\ -0.333 \\ 2.400 \\ 2.200 \end{pmatrix}$	$f_4 := \begin{pmatrix} -0.333 \\ -1.667 \\ 1.500 \\ -0.500 \end{pmatrix}$
2.	$f_1 := \begin{pmatrix} 0.250 \\ 0.333 \\ 0.200 \\ 0.100 \end{pmatrix}$	$f_2 := \begin{pmatrix} 0.330 \\ 0.250 \\ 0.167 \\ 0.143 \end{pmatrix}$	$f_3 := \begin{pmatrix} 0.500 \\ 1.250 \\ 0.800 \\ 1.143 \end{pmatrix}$	$f_4 := \begin{pmatrix} -0.667 \\ 1.333 \\ 1.250 \\ -0.750 \end{pmatrix}$
3.	$f_1 := \begin{pmatrix} 3 \\ 4 \\ 2.4 \\ 1.2 \end{pmatrix}$	$f_2 := \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1.714 \end{pmatrix}$	$f_3 := \begin{pmatrix} 4 \\ 3 \\ 4.4 \\ 4.2 \end{pmatrix}$	$f_4 := \begin{pmatrix} 3 \\ 5 \\ 4 \\ 2 \end{pmatrix}$
4.	$f_1 := \begin{pmatrix} 2 \\ 2.667 \\ 1.6 \\ 0.8 \end{pmatrix}$	$f_2 := \begin{pmatrix} 2.667 \\ 2. \\ 1.333 \\ 1.143 \end{pmatrix}$	$f_3 := \begin{pmatrix} 3 \\ 1.667 \\ 3.6 \\ 3.8 \end{pmatrix}$	$f_4 := \begin{pmatrix} 1.667 \\ 3.667 \\ 3 \\ 1 \end{pmatrix}$
5.	$f_1 := \begin{pmatrix} 3.25 \\ 4.333 \\ 2.6 \\ 1.3 \end{pmatrix}$	$f_2 := \begin{pmatrix} 4.333 \\ 3.25 \\ 2.167 \\ 1.857 \end{pmatrix}$	$f_3 := \begin{pmatrix} 4.25 \\ 3.333 \\ 4.6 \\ 4.3 \end{pmatrix}$	$f_4 := \begin{pmatrix} 3.333 \\ 5.333 \\ 4.25 \\ 2.25 \end{pmatrix}$

Задание 3.

1. Вычислите определитель матрицы с помощью встроенных функций:

$$M := \begin{pmatrix} -1.6 & 5.4 & -7.7 & 3.1 \\ 8.2 & 1.4 & -2.3 & 0.2 \\ 5.3 & -5.9 & 2.7 & -7.9 \\ 0.7 & 1.9 & -8.5 & 4.8 \end{pmatrix}.$$

2. Вычислите определитель матрицы по схеме Гаусса с точностью до 0,0001:

$$M := \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}.$$

Самостоятельная работа 6

Решение алгебраических уравнений и систем

Задание 1. Найти все корни заданного алгебраического уравнения, используя функции *root* и *polyroots*. Сравнить полученные, решения.

Порядок выполнения задания:

1.Задать функцию, нули которой требуется найти.

2.Построить график функции для локализации ее нулей.

3.Задать начальное приближение.

4.Найти и вывести полученное значение вещественного корня.

5.Для поиска комплексных корней определить мнимую единицу.
6.Задать комплексное начальное приближение.
7.Последовательно найти все остальные корни, исключая найденные корни из исходной функции.
8.Задать вектор коэффициентов многочлена.
9.Воспользоваться функцией polyroots для поиска корней.

Варианты:

1. $x^5 - 2.2x^3 + 0.5x^2 - 7x = 3\ 4$
2. $x^5 - 3.2x^3 + 1.5x^2 - 7x = 5.4$
3. $x^5 - 5.2x^3 + 2.5x^2 - 7x = 2.4$
4. $x^5 - 4.2x^3 + 3.5x^2 - 7x = 7.4$
5. $x^5 - 2.2x^3 + 7.5x^2 - 7x = 3.9$
6. $x^5 - 2.9x^3 + 6.5x^2 - 7x = 5.4$
7. $x^5 - 3.2x^3 + 9.5x^2 - 7x = 7.5$

Задание 2. Решить систему из двух нелинейных уравнений.

Порядок выполнения задания:

1.Задать начальное приближение.
2.Открыть блок решения, набрав слово Given.
3.Определить все уравнения системы, используя знак жирного равенства.
4.Найти решение системы, используя функцию Find.
5.Проверить полученное решение.

Варианты:

1. $\sqrt{2x + y + 2} = 3$ и $\sqrt{x + 2y + 5} = y - x$
2. $\sqrt{y - x + 5} = 3$ и $\sqrt{x + y - 5} = 11 - 2y$
3. $\sqrt{2x + y + 1} - \sqrt{x + y} = 1$ и $3x + 2y = 4$
4. $\sqrt{x - y} + \sqrt{x + y} = 3$ и $x + y = 4$
5. $\sqrt{3y - x + 8} = 4$ и $\sqrt{2y - x + 4} = 6x - 3$
6. $\sqrt{x + y + 1} = 3$ и $\sqrt{3x - y + 8} = 2y - 12$

Задание 3. Найти точку пересечения прямой и плоскости, построить графики.

Варианты:

1. $\frac{x - 1}{2} = \frac{y - 1}{-1} = \frac{z - 8}{2}$ и $4x + 2y - z - 11 = 0$
2. $\frac{x - 3}{1} = \frac{y + 2}{-1} = \frac{z + 2}{3}$ и $5x + 9y + 4z - 25 = 0$
3. $\frac{x - 7}{3} = \frac{y - 3}{1} = \frac{z + 1}{-2}$ и $2x + y + 7z - 3 = 0$

Самостоятельная работа 7: Аппроксимация функций

Задание 1.

Варианты:

1.Независимая переменная задана с шагом 0,15 для 20 значений, начиная с x=-0.8. Зависимая переменная задана при помощи генератора случайных чисел – rnd(10). Провести глобальную интерполяцию полученных данных и построить график интерполяции.

2.Проверить совпадение интерполяционных полиномов g(z) и L(z) для всех точек заданного диапазона переменной z.

3.Составить алгоритм локальной линейной интерполяции исходных данных и рассчитать меру отклонения линейной интерполяции от глобальной по следующей формуле: $S = \sum_p (L(x_p) - l(x_p))^2$, где l(x) – линейная интерполяция.

4.Независимая переменная задана с шагом h=0,15 для n=20 значений. Зависимая переменная задана при помощи генератора случайных чисел по формуле $y = h \cdot i^2 + rnd(10)$. Составить алгоритм сглаживания данных.

5.Составить алгоритм сплайн-интерполяции для таблицы экспериментальных данных, в которой x меняется с шагом 0.2 от 0 до 1, а вектор y создается генератором случайных чисел rnd(1).

Задание 2. По таблице значений проинтерполировать табличную функцию тремя способами, построить графики интерполирующих функций.

Порядок выполнения задания:

1.Ввести таблицу значений в виде двух столбцов V_x и V_y, первый из которых содержит значения аргументов, а второй – значения интерполируемой функции.

2.Определить число точек в наборах данных с помощью функции length.

3.Определить линейную интерполяционную функцию Ylin(x) с помощью системной функции lineterp.

4.Определить интерполирующую функцию f(x) в форме полинома Лагранжа.

5.Задать множество значений аргумента для построения графиков функций.

6.Построить на одном рисунке графики полученных функций.

7.Вывести два значения полученных функций: одно в узле, другое вне узлов.

8.Реализовать сплайн-интерполяцию с построением сплайн-функции и выводом значений в тех же точках.

Варианты:

№	i	1	2	3	4	5	6	7
	x_i	1.20	1.50	1.75	2.15	2.55	2.75	3.00
1	y_i	-0.670	-0.065	0.177	0.606	1.154	1.221	2.308
2	y_i	-9.56	-9.33	-9.11	-9.02	-8.71	-8.63	-8.57
3	y_i	1.261	1.280	1.291	1/306	1.321	1.336	1.352
4	y_i	-0.860	-0.818	-0.779	-0.641	-0.504	-0.370	-0.137
5	y_i	33.11	34.81	36.59	38.47	40.44	37.52	34.70

Задание 3. По заданной таблице значений найти регрессионные коэффициенты, построить график линейной регрессии и определить точку максимального отклонения.

Порядок выполнения задания:

1.Ввести таблицу значений в виде двух столбцов V_x и V_y, первый из которых содержит значения аргументов, а второй – экспериментальные значения.

2.Определить число точек в наборах данных с помощью функции rows.

3.Вычислить регрессионные коэффициенты с помощью функций intercept и slope.

4.Определить регрессионную линейную функцию y(x).

5.Задать ранжированную переменную i, принимающую значения от 0 до n.

6.Найти максимальное по абсолютной величине отклонение от линейной регрессии.

7.Найти номер точки максимального отклонения imax.

8.Построить график линейной регрессии совместно с экспериментальными значениями, отметив точку максимального отклонения.

Варианты:

№	i	1	2	3	4	5	6	7	8	9	10
	x_i	-2.0	-1.5	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5
1	y_i	11.3	13.2	14.1	14.1	13.0	10.9	7.9	3.8	-1.3	-7.4
2	y_i	-3.2	-1.8	-0.4	0.9	2.3	3.6	5.0	6.4	9.1	11.3
3	y_i	11.0	8.5	6.7	5.2	4.0	3.1	2.4	1.9	1.5	1.2
4	y_i	-20.5	-16.2	-12.3	-8.9	-6.0	-3.6	-1.7	-0.3	0.6	1.0
5	y_i	1.1	1.3	1.5	1.6	1.7	1.7	1.8	1.9	1.9	2.0

Самостоятельная работа 8: Решение дифференциальных уравнений

Задание 1. Решить на отрезке $[x_0, x_{end}]$ задачу Коши для уравнения первого порядка с постоянным шагом. Получить и изобразить графики решений, вычисленных с шагами h и 4h. Сравнить с точным решением.

Порядок выполнения задания:

1.Задать начальное значение функции как элемент вектора (т.е. в виде переменной с нулевым значением индекса).

2.Создать функцию D(x,y), которая вычисляет значение переменной при заданных значениях зависимой переменной и неизвестной функции.

3.Определить начальное и конечное значения отрезка интегрирования.

4.Указать число шагов интегрирования.

5.Вычислить численное решение при помощи функции rkfixed: z:= rkfixed (y,a,b,N,D). Просмотреть результат вычислений – матрицу z с двумя столбцами, первый из которых содержит значения независимой переменной, а второй соответствующие значения функции.

6.Вычислить численное, решение при учетверенном числе разбиений.

7.Создать функцию для вычисления точного решения.

8.Построить графики приближенных (двух) и точного решения.

9.Изменить число шагов N и проследить за решениями.

Варианты:

№	Уравнение	X_0	X_{end}	$Y(x_0)$	Точное решение
1	$y'+ytgx=sec(x)$	0	1.5	1	$y=sin(x)+cos(x)$
2	$x^2 y'+xy+1=0$	1	3	1	$xy=1-ln\|x\|$
3	$(2x+1) y'=4x+2y$	0	4	1	$y=(2x+1)ln\|2x+1\|+1$
4	$x(y'-y)=e^x$	1	3	e	$y=e^x(ln\|x\|+1)$
5	$y=x(y'-x\ cos(x))$	$\pi/2$	2π	$\pi/2$	$y=x\ sin(x)$
6	$y'=2x(x^2+y)$	1	2	e	$y=e^{x\wedge 2}-x^2-1$
7	$(x\ y'-1)lnx=2y$	1	3	0	$y=ln^2x-lnx$

Задание 2. Решить задачу Коши $y'_1=f_1(x,y_1,y_2)$, $y'_2=f_2(x,y_1,y_2)$, $y_1(a)=y_{0,1}$, $y_2(a)=y_{0,2}$ на отрезке [a,b] с постоянным шагом h=0.1. Изобразить графики решений, вычисленных с шагами h и 2h.

Варианты:

№	$f_1(x,y_1,y_2)$	$f_2(x,y_1,y_2)$	$y_{0,1}$	$y_{0,2}$	a	b
1	$x^2y_1+y_2$	$cos(y_1+xy_2)$	-1	1	0	4
2	$x/(1+x^2+y_2^2)^{1/2}$	$Y_2/(2+x^2+y_1^2)^{1/2}$	1	0	0	5
3	$sin(x^2+y_2^2)$	$cos(xy_1)$	0.2	0	-1	1
4	$sin\ y_2$	$cos\ y_1$	0	0	0	4
5	$x\ cos(y_1+y_2)$	$sin\ (y_1-y_2)$	0.5	-0.5	-1	3
6	$sin\ y_1\ cos^2y_2$	$cos\ y_1\ cos\ y_2$	-0.6	2	2	5
7	$2(3x^2+y_1^2+y_1)^{1/2}$	$(x^2+y_1^2+y_1)^{1/2}$	0	0	-1	3

Самостоятельная работа 9: Программирование

Задание 1. Реализовать программный блок вычисления суммы ряда с точностью 10^{-4}. Вывести на печать значение суммы и число членов ряда, вошедших в сумму. Результат проверить с помощью панели инструментов Calculus.

Варианты:

1. $S = \frac{\pi}{6} - \frac{(\frac{\pi}{6})^3}{3!} + \frac{(\frac{\pi}{6})^5}{5!} - \ldots + (-1)^n \frac{(\frac{\pi}{6})^{2n+1}}{(2n+1)!}$;
2. $S = \frac{2}{3}\sin 2x - \frac{3}{8}\sin 3x - \ldots + (-1)^n \frac{n}{n^2-1}\sin nx$;
3. $S = 1 + \frac{x^2}{2!} - \frac{3x^4}{4!} - \ldots + (-1)^{n+1} \frac{2n-1}{(2n)!}x^{2n}$;
4. $S = \frac{1}{x} - \frac{1}{3x^3} + \frac{1}{5x^5} - \ldots + (-1)^n \frac{1}{(2n+1)x^{2n+1}}$.

Задание 2. Найдите корень уравнения методом половинного деления на заданном отрезке с точностью 10^{-5}. Определить число итераций. Сравнить полученные результаты с результатами, выдаваемыми встроенной функций root.

Варианты:

1	$1 - \sin(x) - \ln(1+x) = 0$	[0,1.5]
2	$x - \frac{1}{3+\sin(3.6x)} = 0$	[0,0.85]
3	$\ln(\lvert x\rvert + 1)\cdot\cos(x) - x^3 = 0$	[0.5,1.5]
4	$x^2 + 10x - 10 = 0$	[0,1]
5	$\sin(x) + \cos(x) - 1.78e^{x-3} = 0$	[1,2]
6	$e^x - x^2 + 12 = 0$	[-5,-3]

Задание 3. Вычислить интеграл методами прямоугольников и трапеций на заданном отрезке, разбив его на 100 частей. Найти точное решение и относительные погрешности для каждого из методов. Сделать соответствующие выводы.

Варианты:

1	$\int \frac{1}{x^2}\cdot\sin\frac{1}{x}dx$	[1,2.5]
2	$\int x^x(1+\ln x)dx$	[1,3]
3	$\int \frac{1}{(25+x^2)\sqrt{25+x^2}}$	[0,5]
4	$\int x^2\sqrt{16-x^2}$	[0,4]
5	$\int \frac{x^4}{(2-x^2)^{\frac{3}{2}}}$	[0,1]

6. НЕОБХОДИМОЕ ТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Лекции и практические занятия проводятся в стандартной аудитории, оснащенной в соответствии с требованиями преподавания теоретических дисциплин.

Для проведения лабораторных работ необходимо:

1. Компьютеры класса Pentium (10 посадочных мест), сетевое окружение.
2. Интернет-центр.
3. Системное программное обеспечение.
4. Среда программирования Турбо Паскаль, Delphi.
5. Прикладное программное обеспечение.

7. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Лекционные занятия по дисциплине "Компьютерные науки" для специальности 010101 – "Математика" проводят:

1) 1, 2 семестр – старший преподаватель кафедры математического анализа и моделирования Салмашова Е.М.;
2) 3, 4 семестр – ассистент кафедры математического анализа и моделирования Кузьменко В.А.;

Лабораторные занятия проводят:

1) 1, 2 семестр – старший преподаватель кафедры математического анализа и моделирования Салмашова Е.М.;
2) 3, 4 семестр – ассистент кафедры математического анализа и моделирования Кузьменко В.А.;
3) 5, 6 семестр – старший преподаватель кафедры математического анализа и моделирования, к.т.н. Рыженко А.В.